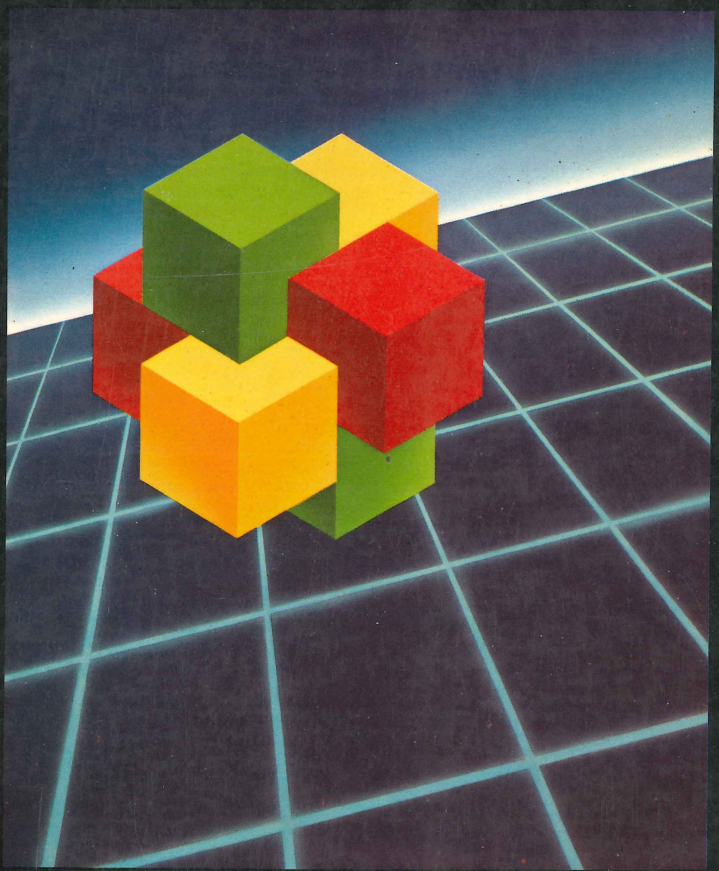
PANEL '85
EXPODATA

20 a 27 de Julho de 1985
UFRGS - Porto Alegre - Brasil

V Congresso da Sociedade Brasileira de Computação
XI Conferência Latino-Americana de Informática



ANAIS vol. II

II SBCCI

V CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO
XI CONFERÊNCIA LATINO-AMERICANA DE INFORMÁTICA

Porto Alegre , Brasil, 20 a 27 de julho de 1985

II SIMPÓSIO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS

A N A I S

Volume II

Editores: Ricardo A. L. Reis & Carlos E. Rech

Promoção:
SBC: Sociedade Brasileira de Computação
CLEI: Centro Latino-Americano de Estudos em Informática
UFRGS: Universidade Federal do Rio Grande do Sul

Patrocínio: FINEP, CAPES, CNPQ,SEI, IFIP, UNESCO,IBI,CREI, FIPEC

Apoio: ITAUTEC, SCOPUS

COMISSÃO DO II SIMPÓSIO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS

Coordenador: Ricardo Augusto da Luz Reis

Altamiro Amadeu Suzim	CPGCC/UFRGS
Ana Maria Kuniyoshi	IM/CTI
Antonio Carlos Barbosa	ITAUCOM
Carlos Ignácio Mammana	IM/CTI
Eber Assis Schmitt	NCE/UFRJ
Jean Albert Bodinaud	IME/USP
Mario Vaz da Silva Filho	DEE/UFRJ
Victor Blatt	CPqD/TELEBRAS

APRESENTAÇÃO

O II Simpósio de Concepção de Circuitos Integrados, evento integrante do V Congresso da Sociedade Brasileira de Computação e XI Conferência Latino-Americana de Informática tem por objetivo promover a troca de experiências, o debate e a busca de uma ação coordenada na área de concepção de circuitos integrados. O sucesso no desenvolvimento de uma tecnologia própria em microeletrônica viabilizará a indústria de eletrônica e de informática, evitando a exportação de empregos especializados e a dependência tecnológica.

No I Simpósio de Concepção de Circuitos Integrados foram apresentados 10 trabalhos técnicos. No II SBCCI serão apresentados 25 trabalhos técnicos, além de tutoriais e palestras especiais. Este volume contêm os 25 trabalhos técnicos e o tutorial "VLSI Architecture".

Não apenas aumentou o número de trabalhos submetidos, mas houve uma evolução muito significativa na qualidade científica dos mesmos.

Este II SBCCI faz parte das atividades da Comissão Especial em Concepção de Circuitos Integrados da Sociedade Brasileira de Computação.

Ricardo Augusto da Luz Reis
Coordenador do II SBCCI
Coordenador da Comissão Especial em
Concepção de Circuitos Integrados.

INDICE DOS TRABALHOS

SESSÃO I

SESSÃO II

M. Nussbaum*

RESUMEN

Se propone una arquitectura sistólica para el problema de ordenamiento. El circuito ordenador propuesto se basa en un algoritmo de enumeración, consistente en dos fases: 'ranking' y reordenamiento. 'Ranking' se realiza utilizando un arreglo sistólico, cuyo comportamiento AT^2 es $O(n^3)$. El reordenamiento es posible de realizar en tiempo constante ocupando una red $O(n^2)$, dando así un comportamiento area tiempo de $O(n^3)$.

ABSTRACT

A sistolic architecture for sorting is proposed. The sorting circuit is based on an enumeration algorithm, which consists of two phases: ranking and rearanging. Ranking is done using a sistolic array whose AT^2 is $O(n^3)$. Rearanging is possible to do in constant time using an $O(n^2)$ network. In this form the sorter area time behaviour is $O(n^3)$.

* Ingeniero Civil Electricista (PUC 1980); M.Sc. Information and Computer Science (GA TECH, 1984). Profesor Escuela de Ingenieria, Departamento Ciencia de la Computación (143), P. Universidad Católica de Chile, Casilla 6177, Santiago, Chile. Investigación financiada parcialmente por DIUC.

Sorting is one of the main activities in data processing. Any application that manages a file system entails sorting. Therefore, it is important to design a fast and cost effective system to handle sorting applications. Such a system could be used as a standalone, or as a special purpose processor in connection with a central processing unit.

In order to implement such a device, it is necessary to achieve a high degree of integration. A new generation of logic design has evolved with VLSI. This new technology makes possible the construction of a collection of processors organized in a regular topology in a single chip. What before was a set of chips with one processor per chip now includes many processors per chip [1].

Since a mapping of high level computations into hardware structures is possible [2], the VLSI approach could present a powerful yet cost effective solution to the sorting problem. However, a number of constraints must be considered. These constraints include limitations in the I/O bandwith, interprocessor communication, and data routing overhead.

A high degree of throughput can be achieved in sorting by introducing systolic architecture. As the name implies, in a systolic architecture input data streams traverse the processor array in a synchronous fashion, passing through different processing elements before reaching their output. A systolic architecture is easy to implement

not only because it has a regular computing pattern, but also because it is easy to reconfigure. A variety of outside constraints can be met, balancing the available I/O bandwith with the host computer. This permits a sustained continuous activity of the host and the coprocessor [3],[4].

The underlying algorithm used will decide the VLSI processor structure and the degree of parallelism and pipelinig. The amount of processing power one can put on a chip will depend (besides the technology) on the size of the processor and the amount of communication required between processors. Therefore, one must employ a simple algorithm that supports a high degree of concurency with minimum amounts of communication between processors.

From the study of the existing sorting algorithms, one finds that most algorithms have one or more of the following problems, making them unsuitable for VLSI implementation:

1. Difficulty in processor-memory comunication (memory conflicts) [8], [9].

2. High amount of computation required in each processor [5], [10], [11], [12]

3. High degree of communication between processors [13], [14].

There is an additional point in which most algorithms fail. The stable condition (records with equal keys retain their original relative order [15]) is difficult to achieve, and most algorithms do not treat this point or have to analyze it as a special case [5],[6],[16].

We present a VLSI sorting network that overcomes the problems discussed in the above lines. The proposed network is based on an enumeration sorting algorithm (records with equal keys retain their original relative order [15]), [1], [20], [21]. It consists of two phases: ranking and rearanging. Ranking consists of finding the rank of the elements in the group of numbers to be sorted. Rearanging places each key in its final position according to its rank.

Ranking is done using a systolic array. The area time complexity (AT^2) of the systolic array is $O(n^3)$ [18] using Thompson's model [5]. But Thompsons's modell does not take into account the complexity of the processing unit. In comparing two circuits, both having the same number of nodes, same number of steps to sort, and same degree of concurrency, the model will give the same area time complexity, even if the processing units in one circuit are P times more complex than those used in the other one. Different processing unit complexities yield different areas per node and different times per step (more processing time and more communication between processors), which in practical terms, lead to different area time behaviors. Therefore, further considerations are required to compare various sorting circuits at implementation level.

Rearanging may be done in constant time at the systolic array output, using an $O(n^2)$ network (e.g. crossbar switch). So the AT^2 of the sorting configuration will be $O(n^3)$.

Figure 1 shows the general scheme used for sorting. As a pipeline machine, series of sequences of numbers may be pumped at the input of the systolic ranking machine, while at its output the sequences appear with its sorted indexes. This information goes to the rearranger, i.e. an alignment network [19], which takes the indexes asociated to each number of the sequence, and put the number to its sorted place.

SYSTOLIC SORTER DESIGN

Consider a matrix of NxN processors for sorting N records. The records to be sorted are presented to the columns and the rows of processors. Each processor has a comparator performing the operation:

$$C_{ij} = (X_i \geq X_j)$$

So the rank of the i^{th} record, R_i, is defined as:

$$R_i = \sum_{j=1}^{N} C_{ij} \qquad (1)$$

Figure 2 shows the rank computation using the above definitios for N=4 and the sequence (6,2,4,3). A similar approach is presented by a

N^2-processor-Rank sort [5]. Both approaches, however, are not stable sorting algorithms. In this case records with identical key (i.e., identical rank) present problems in the rearanging process, because the final position of the record, uniquely determined by the rank of that record, will be shared by other records with the same rank (see Figure 3).

One simple solution to this problem is to use the fact that the lower triangle is the transpose of the upper triangle, in the way the operations are defined, i.e. $C_{ij}=C'_{ji}$. Also $C_{ii}=0$ for all i, so:

$$R_i = \sum_{j=1}^{i-1} C_{ij} + \sum_{j=i+1}^{N} C'_{ji} \qquad (2)$$

Hence stability is mantained without using special conditions over the ones defined for sorting, which is very difficult to achieve [6], [10]. The approach in which all processors are not performing the same operation presents difficulties, especially in VLSI implementation. For example [6] has to use control signals to tell the processors which one of the two operations to perform.

So it will be sufficient to use only the lower triangular part of the matrix of processors to calculate the rank (Figure 4). This approach may be easily implemented in VLSI. Based on the lower triangular matrix, a systolic array with triangular topology can be constructed. From eq.(2), the rank of the i^{th} record is the sum of two terms: one is the sum of the comparison results on the i^{th} row; and the other is the sum of the comparison results on the i^{th} column of the lower triangular matrix. Figure 5 shows a sorter of size 4. The inputs are records to be sorted

while the outputs are records with their corresponding rank. The inputs, with skewed pattern (Figure 5a), are fed into the sorter both horizontally and vertically. Furthermore, each input record has a counter associated with it. The counter is used to calculate the rank of the record. A counter, with initial value zero called horizontal counter H_i or vertical counter V_i, depending upon the corresponding record (X_i) is fed to the sorter horizontally or vertically.

From the above description it is easy to visualize that the hardware required to implement each processor is extremly simple.

There are two basic processing units. Units represented by the square (Figure 5a) have two inputs; one horizontal input record (X_i) with a horizontal counter (H_i) and a vertical input record (X_j) with a vertical counter (V_j). The processor performs the operation $C_{ij}=(X_i \geq X_j)$, which updates the horizontal counter. The complement of C_{ij}, C'_{ij}, is used to update the vertical counter (V_j). The horizontal output of a square unit is the horizontal record and its updated counter (H_{ij}). Similarly, the vertical output is the vertical record (X_j) and its updated counter (V_j). Figure 5c represents the logic design of the processor.

Units represented by the circle in Figure 5d perform the addition of the horizontal and vertical counter i, whose output is the rank of the record.

From the figures, it can be seen that each processor is independent of the other and that only a synchronization signal must go through all of them. this allows a systolic approach to the problem, which offers a high-degree of parallelism and pipelining. Figure 6 shows a pipelining aplication, where three sequences are sorted.

FINAL REMARKS

There are cases in which the sequence to be sorted is bigger than the capacity of the sorting chip, i.e. when m, the size of the capacity of the sorting circuit is less than n, the length of the sequence to be sorted. There are several approaches to this problem [18]. One of the ways to solve this problem is using the inherent modularity of the systolic architectures, i.e. extending the capacity of the sorter adding more sorting modules.

Figure 7a shows that any systolic sorter may be decomposed in a triangular and a square sorter. A triangular sorter is the one defined in the previous section, while a square sorter (Figure 7b) is the same one with the only difference that no additions are made between rows and columns. So the two basic construction blocks which may be rearranged as in Figure 7, may sort any sequence of numbers bigger than the basic triangular sorter size.

In order to sort sequences of n=k*m elements, it will be necessary k triangular sorting modules and k(k+1) square sorting modules. When only one sequence is to be sorted, less hardware is required if we want to

maximize the hardware utilization. In this case, only k-1 square sorters are required. This idea can be extended to a series of sequences p, which is smaller than n, the maximum concurency of the configuration. The number of square sorters required in this case is:

$$\begin{cases} k-1 & \text{when } \lceil p/m \rceil < k-1 \\ \lceil p/m \rceil * (\lceil p/m \rceil - 1)/2 & \text{when } \lceil p/m \rceil \geq k-1 \end{cases}$$

Besides the modularity, the significant features of the proposed sorting network are pipelining, a high degree of concurrency, simple processing units and a minimum amount of communication. This last point is of main importance in designing VLSI, since the numbers of layers achievable with current technology is limited.

The architectural approach for sorting presented here, i.e. a triangular systolic architecture has also been used for transitive closure [1], matrix triangularization [17], and two dimensional convolution [3]. These aplications suggest that a triangular topology may be used in a wide range of complex functions. The architecture we proposed, as being an extension of a pipeline and a parallel machine, has the following advantages [3]:

1. The design makes multiple use of each data element.
2. The design maximizes concurrency.
3. Information flows in a simple way through the machine, and control is simple.
4. There are only few basic simple processing cells.

12. D.S. Hirschberg, "Fast Parallel Sorting Algorithms", Communications of the ACM, Vol 21, No. 8, Aug 1978, pp 657-661.

13. I.S. Charnaya, "Sorting by Merging in a Parallel Computer", Automatika i Telemekhanika, No. 9, Sep 1981, pp 186-191.

14. C.D. Thompson and H.T. Kung, "Sorting on a Mesh Connected Parallel Computer", Communications of the ACM, Vol 20, No.4, April 1977.

15. D.E. Knuth, "The Art of Computer Programming, Vol 3: Sorting and Searching", Addison Wesley, Reading, Mass. ,1973.

16. K.E. Batcher, "Sorting Networks and their applications", Proc. AFIPS SJCC., Vol 32, 1968, pp 307-314.

17. W.M. Gentleman and H.T. Kung, "Matrix Triangularization by Systolic Arrays", SPIE, Real-Time Signal Processing IV, Vol 298, 1981, pp 19-26.

18. P.Y. Chen and M. Nussbaum, "Triangular Sorters: A VLSI Systolic Architecture for Sorting", Georgia Institute of Technology, Technical Report No. GIT-ICS 84/13.

19. D.J. Kuck, "The Structure of Computers and Computations", John Wiley & Sons, 1978.

20. D.E. Muller, F.P. Preparata, "Bounds to complexities of networks for sorting and for Switching", JACM, pp 195-201, 1975.

21. F.T. Leighton, "New lower bound techniques for VLSI", Proc. of IEEE FOCS Conf., pp 1-12, 1981.

1. J.D. Ullman, "Computational Aspects of VLSI", Computer Science Press, 1984.

2. Y.P. Chan and K.S. Fu, "Matching Parallel Algorithms and Architecture", 1983 International Conference on Parallel Processing, Aug 1983, pp 335-337.

3. H.T.Kung, "Why Systolic Architectures", IEEE Computer, Jan 1982 pp 37-46.

4. P.S. Liu and T.Y. Young "VLSI Arrray Design Under Constraint of Limited I/O Bandwith", IEEE Transactions on Computers, Vol c-32, No.12, Dec 1983, pp 1160-1170.

5. C.D. Thopson, "The VLSI Complexity of Sorting", IEEE Transactions on Computers, Vol. 32, No.12, Dec 1983, pp 1171-1184.

6. H. Yasura, N.Takagi and S. Yajima, "The Parallel Enumeration Sorting Scheme for VLSI", IEEE Transactions on Computers, Vol C-31, No.12, Dec. 1982, pp 1192-1201.

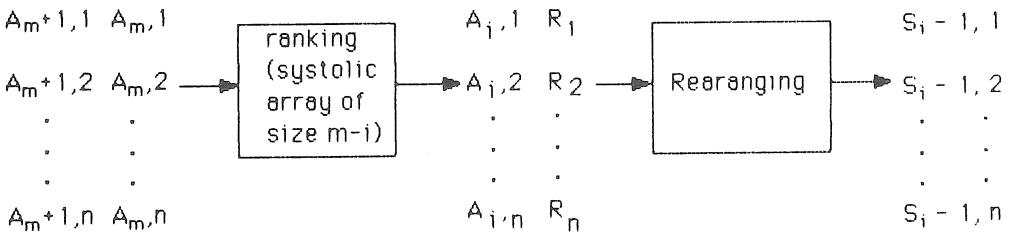
7. P.N. Armstrong and M. Rem, "A serial Sorting Machine", Comput and Elect. Eng. Vol 9, No.1, 1982, pp 53-58.

8. Y. Shiloach and U. Vishkin, "Finding the Maximum, Merging, and Sorting in a Parallel Computation Model", Journal of Algorithms, 2, 1981, pp 88-102.

9. C.P. Kruskal, "Searching, Merging and Sorting in Parallel Computations ", IEEE Transactions on Computers, Vol C-32, No.10, Oct. 1983, pp 942-946.

10. F.P. Preparata, "New Parallel-Sorting Schemes", IEEE Transactions on Computers, Vol C-27, No.7, July 1978, pp 669-673.

11. M. Ajtai, J. Komlos, E. Szemeredi, "An O(nlogn) Sorting Network", Proc 15th ACM Symp. on Theory of Computing, April 1983, pp 1-9.



Used Nomenclature:

$A_{i,j}$: Element j of secuence i to be sorted.
R_i : Rank of record i.
$S_{i,j}$: Element of jth position of the sorted sequence i.

Fig. 1 - General scheme used for sorting.

					Rank
6	1	1	1	1	4
2	0	1	0	0	1
4	0	1	1	1	3
3	0	1	0	1	2
	6	2	4	3	

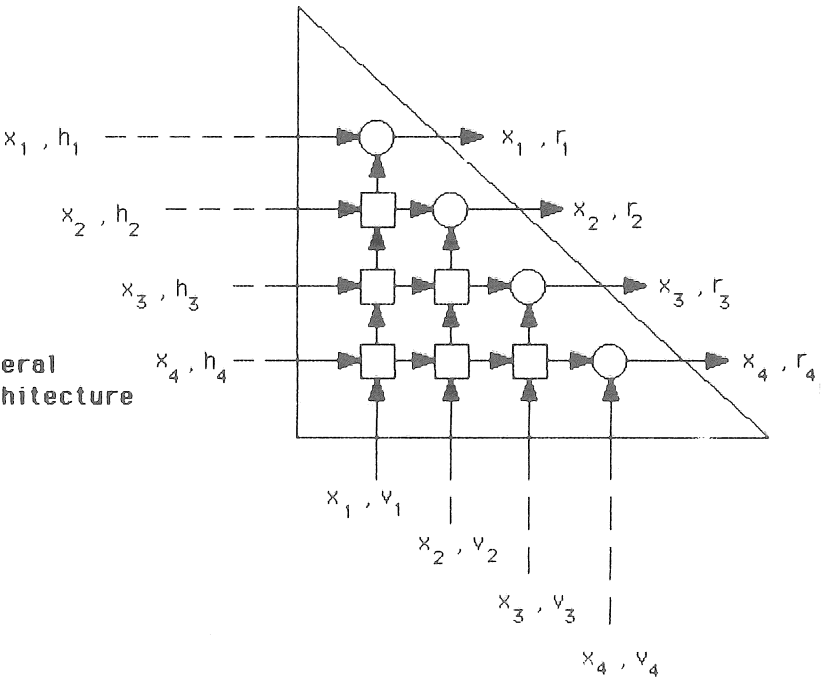
Fig. 2 - Rank Computation

					Rank
6	1	1	1	1	4
2	0	1	0	1	2
4	0	1	1	1	3
3	0	1	0	1	2
	6	3	4	3	

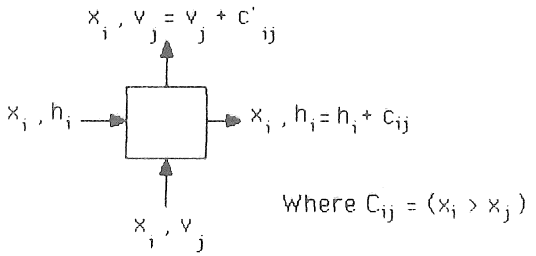
Fig. 3 - Rank Computation With Equal Keys.

6	—	—	—	—
3	0	—	—	—
4	0	1	—	—
3	0	0	0	—
	6	3	4	3

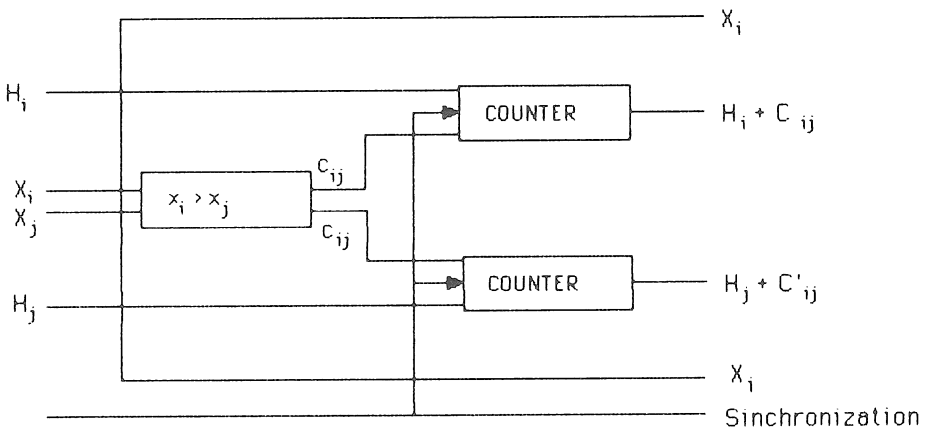
Fig. 4 - Modified Rank Computation.



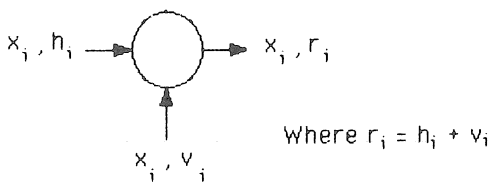
(a) General Architecture



(b) Comparator and Adder



(c) Logic dessign for Comparator and Adder



(d) Adder

Fig. 5 Systolic Architecture for Sorting.

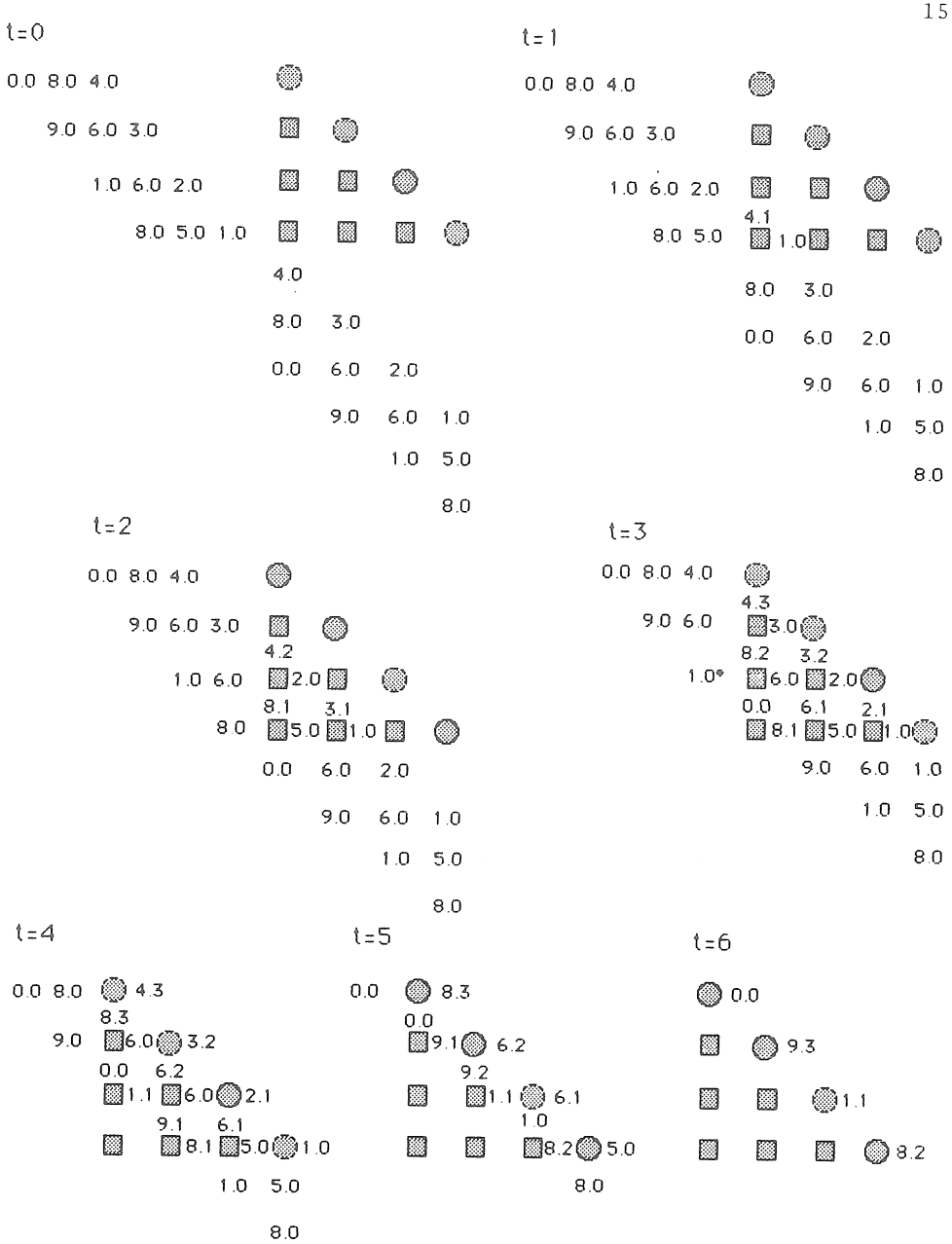


Fig. 6 - Timing secuence for Pipeline Sorting.

S. W. SONG*

SUMÁRIO

Na primeira parte do trabalho serão apresentadas disposições planares compactas para diversos tipos de estruturas de árvore, como árvores binárias com ligações especiais ("threaded trees") e árvores binárias com ligações horizontais ("x-trees"). Na segunda parte serão mostrados resultados de particionamento de árvores que têm aplicação na implementação de sistemas sistólicos em forma de árvore binária ou árvore com ligações especiais, que não cabem em uma única pastilha. Um método será proposto para dispor as pastilhas componentes de maneira eficiente numa placa de circuito impresso.

ABSTRACT

In the first part of this paper we present compact planar layouts for various kinds of tree structures, such as the threaded trees and the x-trees with brother links. In the second part, we show tree partitioning results with application in the implementation of systolic systems with the structure of binary trees or binary threaded trees, that do not fit into one single chip. It will be shown how the component chips can be laid out efficiently on PC boards.

* Engenheiro Eletrônico (EPUSP, 1970), mestre em Matemática Aplicada, modalidade Computação (IMEUSP, 1975), Ph.D. em Ciência de Computação (Carnegie-Mellon University, 1981); Computação Paralela e Computação VLSI; Professor Assistente-Doutor e atual Chefe do Departamento de Matemática Aplicada do Instituto de Matemática e Estatística da USP; IME/USP, Rua do Matão, 1010, Cidade Universitária, CEP 05508, São Paulo, SP; Patrocínio financeiro: CNPq, através da bolsa de pesquisa - processo 300438-82.

1. INTRODUÇÃO

Sistemas sistólicos de alto desempenho, baseados em estruturas hexagonais de elementos de processamento, foram propostos por Kung e Leiserson [5] para computações matriciais. Outras estruturas, como a lista linear, matriz ortogonal e árvore binária, têm sido usadas para diversos outros problemas. A simplicidade e regularidade das interconexões dessas estruturas são propriedades desejáveis para implementação em VLSI [3]. Elas garantem a comunicação local entre elementos vizinhos e uma disposição geralmente compacta na pastilha. Além disso, tais estruturas são modulares, no sentido de permitir a extensão do tamanho da estrutura pela simples duplicação de estruturas menores.

Este trabalho é organizado em duas partes. Na primeira parte, constituída pelas seções 2 e 3, serão propostas disposições compactas planares de diversos tipos de árvore. Nessas disposições, procura-se minimizar a área ocupada e o comprimento das linhas de interconexão entre os elementos ou nós da árvore. Uma aplicação desses resultados é na implementação em pastilha de um sistema sistólico em forma de um dos tipos de árvores apresentados. Os tipos de árvores a serem considerados incluem, além da árvore binária, a árvore binária com folhas ou nós terminais ligados, a estrutura híbrida de árvore e lista linear, a árvore com ligações especiais ou "threads", a x-árvore ou árvore com ligações horizontais e a árvore com ligações especiais múltiplas.

Na segunda parte, seção 4, serão apresentados esquemas de particionamento da árvore binária e árvore binária com ligações especiais em regiões contendo subárvores e a subsequente interligação dessas regiões componentes no plano. Uma aplicação desses resultados é na implementação de um sistema sistólico constituído de uma grande árvore binária, com ou sem ligações especiais, que não cabe numa única pastilha. O particionamento em regiões corresponde à subdivisão do sistema sistólico em pastilhas componentes, que serão dispostas de forma compacta em placas de circuito impresso.

2. ESTRUTURAS DE ÁRVORES CONSIDERADAS

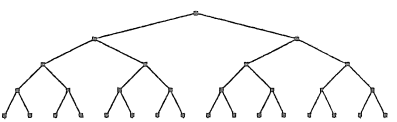


Figura 1

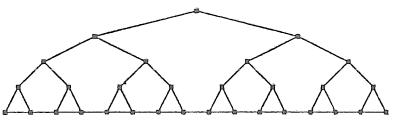


Figura 2

Nesta seção, vamos apresentar os diversos tipos de árvores consideradas. A estrutura da árvore binária da Figura 1 tem sido proposta em vários trabalhos, entre outros, por Bentley e Kung [1], Browning [2] e Song [12]. A Figura 2 mostra uma árvore binária com folhas ou nós terminais ligados. Essa estrutura foi proposta por Leiserson [6] e Magó [8].

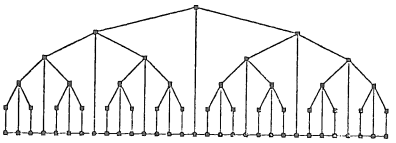


Figura 3

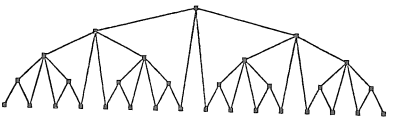


Figura 4

A estrutura híbrida, da Figura 3, é a combinação de uma árvore binária e uma lista linear, com as ligações verticais ligando seus nós. Se denominarmos de n um nó da árvore binária e n' a sua projeção vertical na lista linear, então um percurso em "in-order" (Knuth [4]) dos nós n da árvore binária corresponderá a um percurso dos nós da lista linear da esquerda para a direita. A árvore binária com ligações especiais (ou "threads"), mostrada na Figura 4, foi usada por Rosenberg, Wood e Galil [10], que deram o nome de "dree". A estrutura híbrida foi proposta por Song [13].

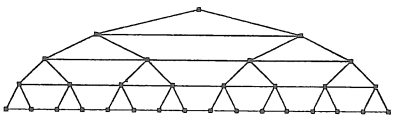


Figura 5

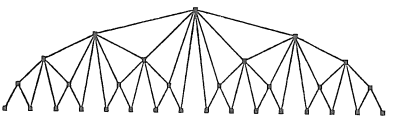


Figura 6

A árvore binária com ligações horizontais, mostrada na Figura 5, também é conhecida pelo nome de x-árvore, conforme o trabalho de Sequin, Despain e Patterson [11]. As ligações horizontais conectam os nós irmãos e primos de um mesmo nível. Finalmente, na Figura 6, é mostrada uma árvore binária com ligações especiais múltiplas.

3. DISPOSIÇÕES COMPACTAS PLANARES

As disposições das Figuras 1 a 6 não são adequadas para implementação em pastilhas. Sendo n o número de nós terminais ou folhas da árvore, a área ocupada naquelas disposições é proporcional a n log n (logaritmo na base 2). A pastilha teria um formato de um retângulo demasiadamente alongado, com um lado proporcional a n e o outro lado proporcional a log n.

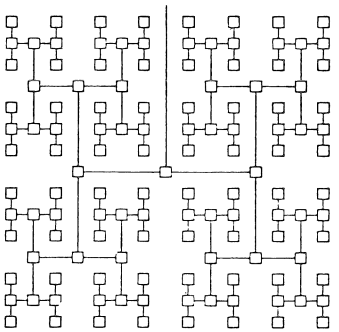


Figura 7

Uma disposição eficiente para a árvore binária da Figura 1 é proposta por Mead e Rem [9]. A chamada disposição-H, mostrada na Figura 7, requer uma área proporcional ao número de nós da árvore. Mostraremos agora como as árvores das Figuras 2 a 6 também podem ser dispostas eficientemente, gozando dessa mesma propriedade.

Mostremos antes que as estruturas das Figuras 2 e 3 são transformáveis à estrutura da Figura 4, e que a estrutura da Figura 5 é transformável à da Figura 6. Imaginemos que as ligações horizontais das Figuras 2, 3 e 5 sejam "elásticas". Se as ligações verticais da Figura 3 começam a encolher de comprimento até atingirem o comprimento nulo, a estrutura resultante será idêntica à da Figura 4. Para a árvore da Figura 2, imaginemos uma pressão aplicada de baixo para cima, elevando as ligações horizontais. Teremos novamente a estrutura resultante idêntica à da Figura 4. Analogamente, a elevação das ligações horizontais da estrutura da Figura 5 dará origem à estrutura da Figura 6. Iremos portanto apresentar disposições planares apenas para a árvore binária com ligações especiais (Figura 4) e a árvore binária com ligações especiais múltiplas (Figura 6), já que as demais são equivalentes a elas.

A árvore binária com ligações especiais da Figura 4 pode ser disposta, como mostra a Figura 8, por um método recursivo. Nesse método, é aplicada uma operação de rotação a partes selecionadas em torno de um eixo apropriado, de tal maneira que as ligações especiais, ou "threads", têm comprimento constante. Tal não se verifica na Figura 4, onde as ligações especiais se tornam mais compridas à medida que elas se aproximam do centro.

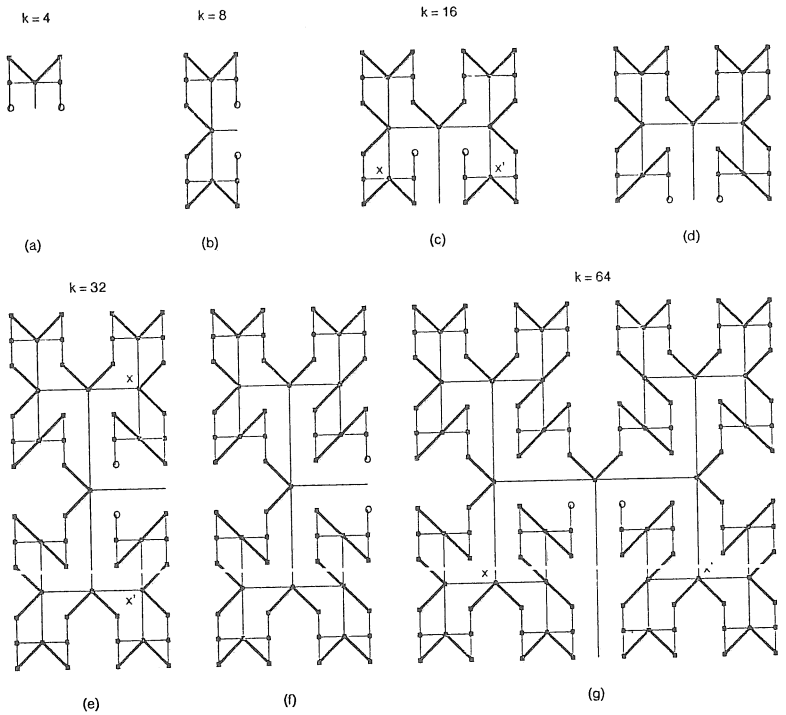


Figura 8

Seja k uma potência de 2, denotando o número de nós terminais de uma árvore binária com ligações especiais. A Figura 8 (a) mostra o caso de k=4. As linhas grossas representam as ligações especiais e os círculos pequenos denotam os dois nós terminais extremos, que não possuem ligações especiais. A Figura 8 (b), para k=8, é obtida combinando-se (a) com a sua imagem especular em relação a um eixo horizontal, e adicionando-se duas ligações especiais a mais. De maneira análoga, a Figura 8 (c), para k=16, é obtida combinando-se (b) com a sua imagem especular em relação a um eixo vertical, e adicionando-se duas ligações especiais.

Para conseguirmos a disposição para k=32, primeiro obtemos (d) a partir de (c), aplicando-se uma rotação de parte de (c) ao longo do eixo x-x', a fim de trazer os nós terminais extremos (pequenos círculos) para perto da periferia. A Figura 8 (e), para k=32, é então obtida combinando-se (d) com a sua imagem especular em relação a um eixo horizontal e adicionando-se mais duas ligações especiais. Para k=64, primeiro obtemos (f) de (e) pela rotação de parte de (e) ao longo do eixo x-x' e depois gerar (g). Assim,

a disposição que resulta tem a propriedade, por construção, de que todas as ligações especiais têm comprimento constante, ligando nós próximos.

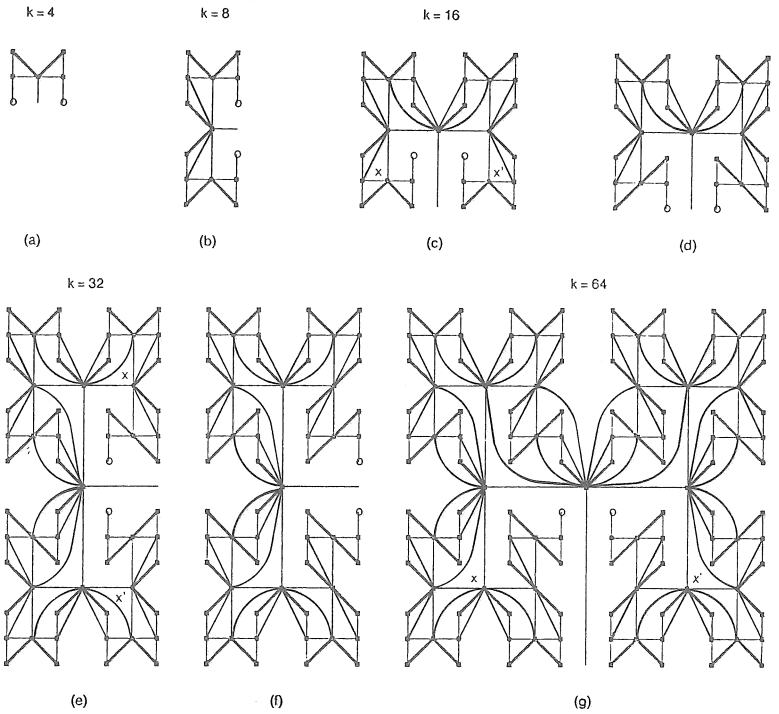


Figura 9

Para obtermos uma disposição compacta para a estrutura de árvore binária com múltiplas ligações especiais, da Figura 6, basta observar que cada árvore parcial desta estrutura, constituída pelos nós desde a raiz até os nós de algum nível, contém a árvore com ligações especiais, do tipo da Figura 4, que já sabemos como dispor. Aplicando assim o mesmo método, teremos a disposição da Figura 9.

4. PARTICIONAMENTO DE ÁRVORES

Consideremos uma estrutura de árvore binária grande, que não cabe numa única pastilha. Queremos particioná-la em regiões ou pastilhas componentes. As propriedades desejáveis num esquema de particionamento de uma estrutura em pastilhas componentes são:

1. Usa poucos tipos diferentes de pastilhas componentes.
2. Usa a área de cada pastilha componente de maneira efetiva (sem desperdício).
3. Permite uma disposição compacta das pastilhas componentes, que serão montadas em placas de circuito impresso .
4. Usa linhas curtas para interligar pastilhas componentes.

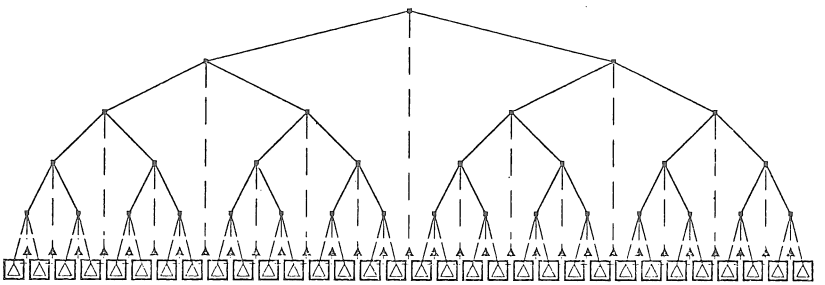


Figura 10

Na Figura 10, cada quadrado representa uma pastilha contendo uma subárvore binária (representada por um triângulo dentro da pastilha). Chamemos tais pastilhas de pastilhas terminais. Para uma árvore binária de k pastilhas terminais, temos exatamente k-1 nós internos da árvore binária que ainda precisam ser alocados em pastilhas. Para acomodar tais nós internos, vamos fazer com que cada pastilha terminal, com exceção da pastilha mais à direita, receba mais um nó, que é o seu sucessor em "in-order", indicado pela linha tracejada da Figura 10. Temos assim a Figura 11.

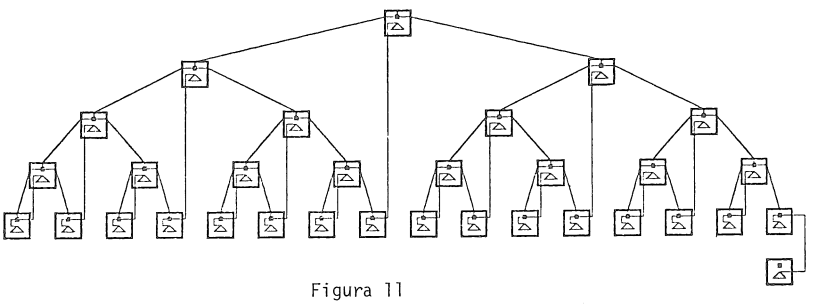


Figura 11

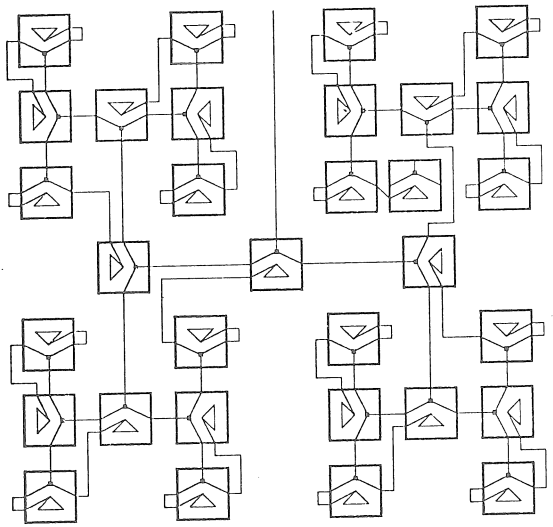


Figura 12

Na Figura 11, conseguimos dispor a árvore binária original em pastilhas componentes, todas de um mesmo tipo. Cada pastilha componente tem sua área usada de maneira efetiva. Como as pastilhas componentes estão dispostas em forma de um tipo particular de árvore binária com ligações especiais (uma só ligação especial ao invés de duas), podemos usar o resultado da seção 3 para produzir a disposição final da Figura 12.

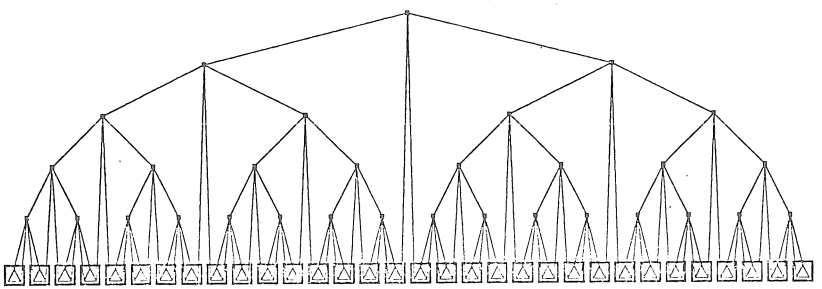


Figura 13

A Figura 13 mostra uma árvore binária com ligações especiais. Novamente, vamos supor que ela deva ser decomposta em pastilhas componentes. Cada quadrado da Figura 13 representa uma pastilha componente, contendo uma subárvore denotada pelo triângulo.

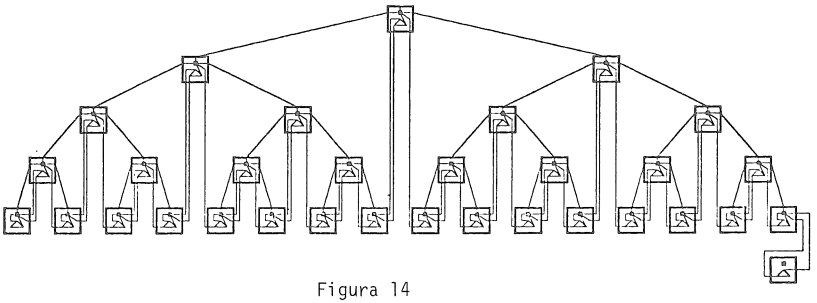


Figura 14

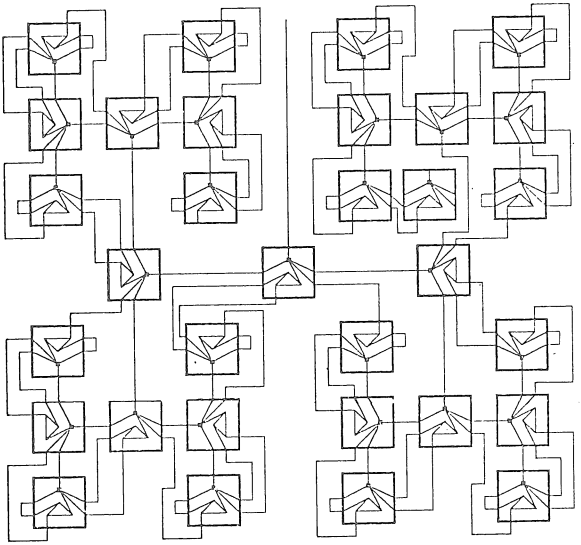


Figura 15

Como antes, fazemos com que cada pastilha terminal receba mais um nó interno, resultando assim na Figura 14. A disposição das pastilhas componentes é exatamente a de uma árvore binária com ligações especiais, onde os nós são as pastilhas. Podemos então apresentar a disposição final da Figura 15, usando novamente resultado da seção 3.

5. CONCLUSÕES

Mostramos na primeira parte deste trabalho que a disposição-H pode ser estendida a diversos outros tipos de estruturas de árvore, onde há ligações especiais, ou ligações horizontais entre nós irmãos e primos. Sendo todas árvores binárias, a idéia de tentar essa extensão não é extraordinária. A importância do trabalho reside principalmente em mostrar que as ligações adicionais podem ser acomodadas dentro da disposição-H de uma forma sistemática e eficiente.

Quanto à segunda parte, a idéia de incorporar um nó interno da árvore binária em cada pastilha terminal é devida a Leiserson [7]. O método proposto aqui difere do dele na escolha do nó a ser combinado, além da utilização da operação de rotação para conseguir a propriedade desejável de comprimento constante nas ligações especiais. No que se refere ao comprimento total das linhas de interconexão entre pastilhas componentes, o método aqui proposto usa essencialmente um quarto do comprimento no método de Leiserson. Além disso, o método proposto é mais geral no sentido de poder estender-se a outros tipos de árvores, como a árvore binária com ligações especiais.

BIBLIOGRAFIA

1. Bentley, J. L., & Kung, H. T., "A Tree Machine for Searching Problems", *Proceedings of 1979 International Conference on Parallel Processing*, pp. 257-266. IEEE. August, 1979.

2. Browning, S. A., "Computations on a Tree of Processors", *Proceedings Conference on Very Large Scale Integration: Architecture, Design, Fabrication*, pp. 453-478. January, 1979.

3. Foster, M. J., & Kung, H. T., " The Design of Special-Purpose VLSI Chips", *Computer*, pp. 26-40. January, 1980.

4. Knuth, D. E., *The Art of Computer Programming. Volume 1: Fundamental Algorithms*, 634 p. Addison-Wesley, 1973.

5. Kung, H. T., & Leiserson, C. E., "Systolic Arrays for VLSI", *Sparse Matrix Proceedings*, pp. 256-282. SIAM. 1979.

6. Leiserson, C. E., "Systolic Priority Queues", *Proceedings Conference on Very Large Scale Integration: Architecture, Design, Fabrication*, pp. 199-214. January, 1979.

7. Leiserson, C. E., *Area-Efficient VLSI Computations*, 132 p. Ph. D. Thesis, Carnegie-Mellon University, Computer Science Department. 1981.

8. Magō, G. A., "A Network of Processors to Execute Reduction Languages", *International Journal of Computer and Information Sciences*, pp. 349-385. March, 1979.

9. Mead, C., & Rem, M., "Cost and Performance of VLSI Computing Structures", *IEEE Journal of Solid State Circuits*, pp. 455-462. April, 1979.

10. Rosenberg, A. L., Wood, D., & Galil, Z., "Storage Representations for Tree-like Data Structures", *Mathematical Systems Theory*, pp. 105-130. Volume 13, Number 2, 1979.

11. Sequin, C. H., Despain, A. M., & Patterson, D. A., "Communication in X-trees, a Modular Multiprocessor System", *ACM Proceedings 1978 Annual Conference*, pp. 194-203. December, 1978.

12. Song, S. W., "A Highly Concurrent Tree Machine for Database Applications", *Proceedings of the 1980 International Conference on Parallel Processing*, pp. 259-268. August, 1980.

13. Song, S. W., *On a High-Performance VLSI Solution to Database Problems*, 161 p. Ph. D. Thesis, Carnegie-Mellon University, Computer Science Department. 1981.

MÁQUINA DE ESTADOS FINITOS EM CIRCUITOS INTEGRADOS - IMPLEMENTAÇÃO EM PLA NÃO DETERMINÍSTICA.

H.T. SILVA*

SUMÁRIO

Este trabalho apresenta uma abordagem diferente para a implementação de máquinas de estados finitos em circuitos integrados. A máquina de estados finitos será representada por um autômato finito não determinístico; convencionalmente ela é representada por um autômato finito determinístico.

Para integrar a máquina será utilizada uma PLA. A PLA que implementa a máquina determinística é normalmente obtida a partir de sua tabela de transição de estados, através de técnicas conhecidas de projetos lógicos. A PLA não determinística será obtida diretamente de seu diagrama de estados.

Será realizada uma comparação entre as PLA's para o caso particular de um controlador projetado como parte de um circuito integrado que controla o acesso a uma rede local.

ABSTRACT

This work shows a different implementation of finite state machines in integrated circuits. The finite state machine will be represented by a nondeterministic finite automata; conventionally it is represented by a deterministic finite automata.

The machine will be integrated in a PLA. The PLA that implements the deterministic machine is normally projected from its state transition table using well-known techniques of logic projects. The nondeterministic PLA will be projected directly from its state diagram.

A comparison between these PLA's is made for a particular case of a controller projected as part of an integrated circuit that controls the access of a local network.

* Engenheira eletrônica (UFRJ, 1981); Projeto de Circuitos Integrados Digitais; Núcleo de Computação Eletrônica da UFRJ (NCE/UFRJ), Ilha do Fundão, Cidade Universitária - Rio de Janeiro, RJ, CEP 20001.

INTRODUÇÃO

O comportamento de qualquer sistema pode ser representado em termos de relações matemáticas entre 3 conjuntos de variáveis, as quais descrevem as entradas, as saídas e os estados do sistema. O conjunto de entradas representa as excitações externas que são aplicadas ao sistema para produzir uma mudança no seu comportamento; o conjunto de saídas representa os pontos de observação que caracterizam a mudança do sistema em resposta às entradas. Uma das características básicas de qualquer sistema é que sua saída atual é função não só das entradas atuais mas também das entradas passadas e das saídas. Devido a isto, podemos pensar em um sistema como possuindo uma "memória" que guarda informações acerca do comportamento anterior do sistema. O conjunto de estados, o terceiro conjunto de variáveis, é pois, usado para representar a quantidade de informação armazenada no sistema.

As relações funcionais que descrevem o comportamento de um sistema podem ser representadas por equações diferenciais - integrais e, portanto, as variáveis deste sistema assumem valores contínuos, ou por autômatos, em que as variáveis assumem valores discretos no tempo.

Um autômato pode apresentar um comportamento determinístico ou não determinístico. O autômato é determinístico quando cada transição do sistema faz com que este vá de um estado para outro, de forma que a cada instante o sistema esteja em apenas um de seus possíveis estados; o autômato é não determinístico quando puder se encontrar em vários estados simultaneamente. A descrição formal dos autômatos será mostrada mais adiante.

É necessário explicar aqui que o conceito de não determinismo não significa que o sistema não é bem determinado, ou seja, suas transições são aleatórias. O autômato não determinístico é apenas uma abstração matemática: enquanto no autômato determinístico toda a "memória" do sistema; em um dado instante de tempo, é representada por apenas um de seus estados; no autômato não determinístico a representação de seu passado está contida em um conjunto de estados. Fisicamente, tanto em um como em outro sistema, as medidas efetuadas para caracterizá-los são as mesmas, ou seja, as variáveis de entrada, de saída e de estado são da mesma natureza. A diferença física entre eles está no fato de que o sistema não determinístico aceita paralelismo; vários caminhos podem ser percorridos ao mesmo tempo para se chegar ao resultado final. Durante o processamento das entradas estes caminhos vão sendo abandonados, ficando apenas um deles, que levará ao resultado final.

Embora a diferença entre autômatos determinísticos e não determinísticos seja apenas conceitual, a implementação em "hardware" destes autômatos são bastante diferentes, conforme será mostrado no decorrer do artigo.

MÁQUINAS DE ESTADOS FINITOS EM CIRCUITOS INTEGRADOS

A forma mais simples de um sistema digital síncrono é a máquina sequencial de estados

finitos {BOOT 67}. Embora sistemas complexos possam ser descritos, é geralmente possível que estes possam ser construídos a partir de máquinas sequenciais.

Uma máquina de estados finitos é formada de uma lógica combinacional e unidades de armazenamento (registradores de estados). As variações na implementação desta máquina em circuitos integrados está relacionada com a forma de implementação da lógica combinacional e com a organização dos registradores de estado. A máquina sequencial pode ser realizada através de ROM's, PLA's, geradores de tempo ou combinações destes {OBRE 82}. Trataremos aqui da implementação de máquinas sequenciais de estados finitos em PLA's {MEAD 80} a partir da descrição das mesmas em forma de um autômato finito.

AUTOMATA DETERMINÍSTICO

Um automato M sobre um alfabeto I é um sistema (Q, I, S, q_0, F, Z) definido por {HOPC 69}, {BOOT 67}:

1. Um conjunto de elementos Q denominados estados;
2. Um conjunto finito I denominado alfabeto de entrada;
3. Uma função S de mapeamento de Q x I em Q;
4. Um conjunto finito Z denominado alfabeto de saída;
5. Um estado inicial q_0 em Q;
6. Uma função F de mapeamento de Q x I em Z.

Quando o conjunto de elementos que formam os estados é finito, dizemos que o autômato é finito.

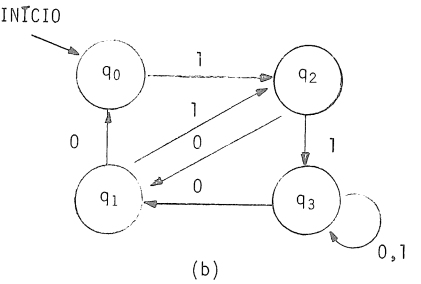
Um autômato finito é representado, para uma melhor visualização, através de um diagrama de estado. Neste diagrama cada estado é representado por um nó e as possíveis transições por linhas direcionadas rotuladas pelas excitações externas que provocam estas transições (variáveis de entrada).

Como exemplo, suponha um autômato finito determinístico que detecta a existência de dois 1's consecutivos ou dois 0's consecutivos em uma cadeia de caracteres. Sua descrição e seu diagrama de estados são apresentados nas figs. 1a e 1b.

$M = (Q,I,S,q_0,F,Z)$
$I = \{0,1\}$
$Z = \{0,1\}$
$Q = \{q_0,q_1,q_2,q_3\}$
$F(q_0,x) = 0$
$F(q_1,x) = 0$
$F(q_2,x) = 0$
$F(q_3,x) = 1$

$S(q_0,0) = q_1$
$S(q_0,1) = q_2$
$S(q_1,0) = q_3$
$S(q_1,1) = q_2$
$S(q_2,0) = q_1$
$S(q_2,1) = q_3$
$S(q_3,0) = q_3$
$S(q_3,1) = q_3$

(a)



(b)

Fig. 1 - Autômato finito determinístico que detecta dois 0's ou dois 1's consecutivos. (a) Descrição e (b) Diagrama de estados.

AUTÔMATO NÃO DETERMINÍSTICO

Um autômato não determinístico M sobre um alfabeto I é um sistema (Q, I, S, q_0, F, Z) definido por {HOPC 69}, {BOOT 67}:

1. Um conjunto de elementos Q denominados estados;
2. Um conjunto finito I denominado alfabeto de entrada;
3. Um conjunto finito Z denominado alfabeto de saída;
4. Uma função S de mapeamento de Q x I em subconjuntos de Q;
5. Um conjunto de estados iniciais em Q;
6. Uma função F de mapeamento de Q x I em Z;

A diferença mais importante entre o autômato determinístico e o não determinístico é que, neste último, S(q,a) é um conjunto de estados (pode ser vazio) ao invés de um único estado. A interpretação de $S(q,a)=\{p_1,p_2,...pk\}$ é que M, no estado q, tendo como entrada a, poderá transicionar para os estados p_1, p_2, ...pk.

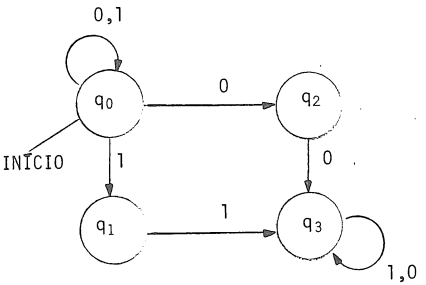
Portanto, o sistema representado por um autômato não determinístico poderá estar em vários estados simultaneamente.

O exemplo anterior, representado por um autômato não determinístico é mostrado nas figuras 2a e 2b.

$M = (Q,I,S,q_0,F,Z)$
$I = \{0,1\}$
$Z = \{0,1\}$
$Q = \{q_0,q_1,q_2,q_3\}$
$F(q_0,x) = 0$
$F(q_1,x) = 0$
$F(q_2,x) = 0$
$F(q_3,x) = 1$

$S(q_0,0)=\{q_0,q_2\}$
$S(q_0,1)=\{q_0,q_1\}$
$S(q_1,0)=\emptyset$
$S(q_1,1)=\{q_3\}$
$S(q_2,0)=\{q_3\}$
$S(q_2,1)=\emptyset$
$S(q_3,0)=\{q_3\}$
$S(q_3,1)=\{q_3\}$

(a)



(b)

Fig. 2 - Autômato finito não determinístico que detecta dois 0's ou dois 1's consecutivos.

(a) Descrição e (b) Diagrama de estados.

CONTROLADOR DA RECEPÇÃO DE PACOTES DA REDE

Para ilustrar a abordagem de PLA determinística e não determinística utilizaremos o exemplo de um controlador. A lógica deste controlador faz parte da lógica de acesso a uma rede local em anel existente no NCE/UFRJ {SILV 83}. Atualmente esta lógica de acesso à rede local está sendo integrada; neste CI, o controlador de recepção de mensagens aqui mencionado como exemplo foi projetado em uma PLA não determinística.

O controlador pode ser representado pelo diagrama em bloco da figura 3.

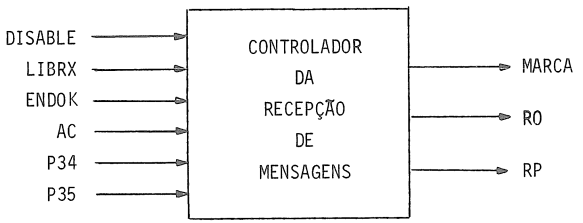


Fig. 3 - Controlador da Recepção de Mensagens - Diagrama em Bloco.

Sua lógica é a seguinte:

Inicialmente o controlador está liberado para receber mensagens da rede. Estas são recebidas na forma de pacotes. O sinal de "DISABLE" coloca sempre o controlado no estado de recepção livre. Quando chega uma mensagem pela rede, é enviado para o controlador o sinal "ENDOK" caso a mensagem seja para a estação deste controlador. A seguir é recebido um sinal "AC" informando se a estação pode aceitar ou não mensagens deste destinatário. O controlador aceitará a mensagem caso não esteja ocupado com a mensagem recebida anteriormente. P34 e P35 significam os tempos do pacote em que devem ser escritos, no próprio pacote recebido, os bits de status, através da saída serial MARCA, avisando à estação remetente do pacote o estado da recepção do mesmo.

Os "status" do pacote são os seguintes:

MARCA		
P34	P35	
0	0	pacote ignorado (a estação provavelmente está desligada)
0	1	pacote aceito pela estação
1	0	estação não foi habilitada para receber deste destinatário
1	1	estação ocupada com a recepção do pacote anterior

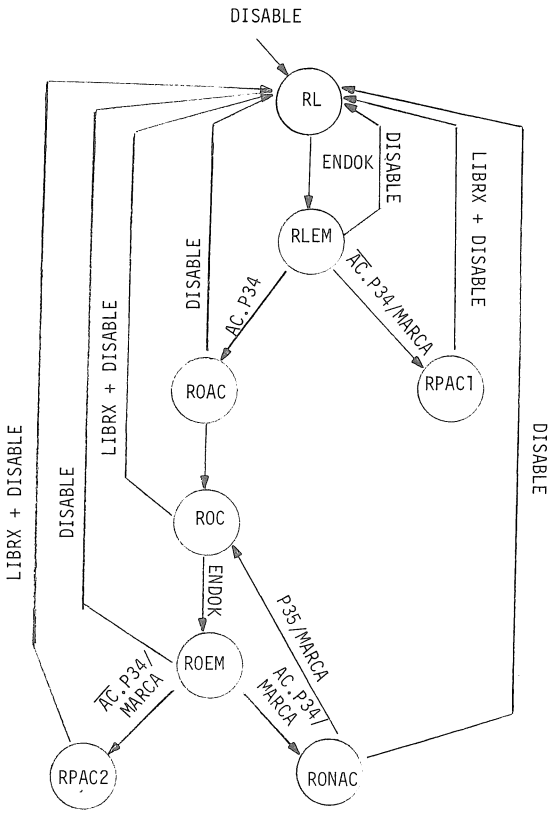
Ao receber um pacote aceito ou rejeitado o controlador gera os sinais "RO" (Recepção Ocupada) ou "RP" (Rejeição Pacote) respectivamente e somente fica apto a receber novos pacotes quando lhe é enviado o sinal "LIBRX", indicando a liberação da estação para nova leitura.

A seguir apresentaremos um autômato determinístico e outro não determinístico que implementam este controlador. O autômato determinístico não corresponde à conversão do não determinístico em determinístico {HOPC 69}. Apenas ambos representam corretamente o controlador em questão.

IMPLEMENTAÇÃO DO CONTROLADOR EM PLA DETERMINÍSTICA

Na fig. 4 temos o diagrama de transição de estados simplificado para o controlador representando uma máquina sequencial de estados finitos determinística. O diagrama apre

sentado encontra-se simplificado, uma vez que devido às dimensões da máquina (6 entradas) seria difícil a manipulação de um diagrama de transição levando-se em consideração todas as possíveis combinações das entradas para a mudança de cada estado da máquina.



Estados:

RL - Recepção Livre;

RLEM - Chegada de Pacote para a Estação que estava Livre;

ROAC - Aceitação de Pacote com Entrada em Recepção Ocupada;

RPAC1 - Rejeição de Pacote em Recepção Livre;

ROC - Recepção Ocupada com Pacote aceito;

ROEM - Chegada de Pacote para a Estação que já estava Ocupada;

RPAC2 - Rejeição de Pacote em Recepção Ocupada;

RONAC - Não aceitação de Pacote devido à Ocupação da Recepção com o Pacote Anterior.

Fig. 4 - Autômato Determinístico do Controlador da Fig. 3.

A codificação para os estados do controlador, escolhida de forma a minimizar as equações da máquina é mostrada na fig. 5.

X1	X2	X3	ESTADO CORRESPONDENTE
0	0	0	RLEM
0	0	1	RL
0	1	0	RONAC
0	1	1	ROC
1	0	0	RPAC1
1	0	1	RPAC2
1	1	0	ROAC
1	1	1	ROEM

Fig. 5 - Codificação dos estados do controlador da fig. 4.

A tabela de transição para o controlador é mostrada na fig. 6.

ENTRADAS						ESTADO ATUAL			PRÓX. ESTADO			SAÍDAS		
DISABLE	LIBRX	ENDOK	AC	P34	P35	X1	X2	X3	X1	X2	X3	MARCA	RO	RP
1	X	X	X	X	X	X	X	X	0	0	1	0	0	0
X	1	X	X	X	X	0	1	1	0	0	1	0	0	0
X	1	X	X	X	X	1	0	0	0	0	1	0	0	0
X	1	X	X	X	X	1	0	1	0	0	1	0	0	0
0	X	1	X	X	X	0	0	1	0	0	0	0	0	0
0	X	0	X	X	X	0	0	1	0	0	1	0	0	0
0	X	X	X	0	X	0	0	0	0	0	0	0	0	0
0	X	X	0	1	X	0	0	0	1	0	0	1	0	1
0	X	X	1	1	X	0	0	0	1	1	0	0	1	0
0	X	X	X	X	0	1	1	0	1	1	0	0	1	0
0	X	X	X	X	1	1	1	0	0	1	1	1	1	0
0	0	0	X	X	X	0	1	1	0	1	1	0	1	0
0	0	1	X	X	X	0	1	1	1	1	1	0	1	0
0	X	X	X	0	X	1	1	1	1	1	1	0	1	0
0	X	X	0	1	X	1	1	1	1	0	1	1	1	1
0	X	X	1	1	X	1	1	1	0	1	0	1	1	0
0	X	X	X	X	0	0	1	0	0	1	0	0	1	0
0	X	X	X	X	1	0	1	0	0	1	1	1	1	0
0	0	X	X	X	X	1	0	0	1	0	0	0	0	1
0	0	X	X	X	X	1	0	1	1	0	1	0	1	1

Fig. 6 - Tabela de Transição para o Controlador da fig. 4.

A partir da tabela de transição de estados apresentada e utilizando-se o método de minimização de funções lógicas conhecido como QUINE-McCLUSKEY {RHYN 73}, chegou-se às seguin

tes equações que descrevem o comportamento da PLA:

X1 = $\overline{\text{DISABLE}}\ \overline{\text{LIBRX}}$ ENDOK $\overline{\text{XT}}$ X2 X3 + $\overline{\text{DISABLE}}\ \overline{\text{AC}}$ P34 X1 X2 X3 + $\overline{\text{DISABLE}}$ P34 $\overline{\text{AC}}\ \overline{\text{XT}}\ \overline{\text{X2}}\ \overline{\text{X3}}$ + $\overline{\text{DISABLE}}\ \overline{\text{P35}}$ X1 X2 $\overline{\text{X3}}$ + $\overline{\text{DISABLE}}\ \overline{\text{P34}}$ X1 X2 X3 + $\overline{\text{DISABLE}}\ \overline{\text{LIBRX}}$ X1 $\overline{\text{X2}}$;

X2 = $\overline{\text{DISABLE}}$ AC P34 $\overline{\text{XT}}\ \overline{\text{X2}}\ \overline{\text{X3}}$ + $\overline{\text{DISABLE}}$ AC P34 X1 X2 X3 + DISABLE LIBRX X1 X2 X3 + $\overline{\text{DISABLE}}\ \overline{\text{LIBRX}}\ \overline{\text{XT}}$ X2 X3 + $\overline{\text{DISABLE}}\ \overline{\text{P34}}$ X1 X2 X3 + $\overline{\text{DISABLE}}$ X2 $\overline{\text{X3}}$ + $\overline{\text{DISABLE}}$ X1 X2 $\overline{\text{X3}}$ + $\overline{\text{DISABLE}}$ P35 X2 $\overline{\text{X3}}$;

X3 = $\overline{\text{DISABLE}}\ \overline{\text{AC}}$ P34 X1 X2 X3 + $\overline{\text{DISABLE}}\ \overline{\text{ENDOK}}\ \overline{\text{XT}}\ \overline{\text{X2}}$ X3 + $\overline{\text{DISABLE}}\ \overline{\text{LIBRX}}\ \overline{\text{XT}}$ X2 X3 + $\overline{\text{DISABLE}}\ \overline{\text{P34}}$ X1 X2 X3 + $\overline{\text{DISABLE}}$ P35 X2 $\overline{\text{X3}}$ + LIBRX $\overline{\text{XT}}$ X2 X3 + LIBRX X1 $\overline{\text{X2}}$ + DISABLE;

MARCA = $\overline{\text{DISABLE}}\ \overline{\text{AC}}$ P34 $\overline{\text{XT}}\ \overline{\text{X2}}\ \overline{\text{X3}}$ + $\overline{\text{DISABLE}}\ \overline{\text{AC}}$ P34 X1 X2 X3 + $\overline{\text{DISABLE}}$ P35 X2 $\overline{\text{X3}}$;

RO = $\overline{\text{DISABLE}}$ AC P34 $\overline{\text{XT}}\ \overline{\text{X2}}\ \overline{\text{X3}}$ + $\overline{\text{DISABLE}}\ \overline{\text{LIBRX}}$ X1 $\overline{\text{X2}}$ X3 + $\overline{\text{DISABLE}}\ \overline{\text{LIBRX}}\ \overline{\text{XT}}$ X2 X3 + $\overline{\text{DISABLE}}$ X2 $\overline{\text{X3}}$ + $\overline{\text{DISABLE}}$ P35 X2 $\overline{\text{X3}}$ + $\overline{\text{DISABLE}}\ \overline{\text{P34}}$ X1 X2 X3;

RP = $\overline{\text{DISABLE}}\ \overline{\text{AC}}$ P34 $\overline{\text{XT}}\ \overline{\text{X2}}\ \overline{\text{X3}}$ + $\overline{\text{DISABLE}}\ \overline{\text{AC}}$ P34 X1 X2 X3 + $\overline{\text{DISABLE}}\ \overline{\text{LIBRX}}$ X1 $\overline{\text{X2}}$.

Por estas equações verificamos que a PLA determinística que implementa o controlador contêm 18 entradas (6 sinais de entrada + 3 estados, com seus complementos), 6 saídas (3 sinais de saída + 3 estados) e 17 termos de produto, correspondendo a uma PLA 24 x 17.

IMPLEMENTAÇÃO DO CONTROLADOR EM PLA NÃO DETERMINÍSTICA

Na fig. 7 temos o autômata que representa o controlador através de uma máquina de estados finitos não determinística.

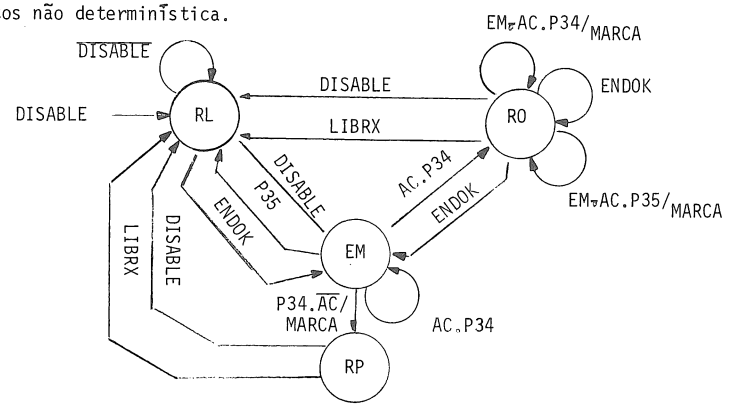


Fig. 7 - Autômato não determinístico do Controlador da fig. 3.

Estados:

RL - Recepção Livre;

RO - Recepção Ocupada;

EM - Chegada de Pacote para a Estação (É pra mim);

RP - Estação não pode receber o Pacote (Rejeição de Pacote)

As equações da PLA para o autômato não determinístico podem ser obtidas diretamente do grafo {FLOYD[6]} da seguinte maneira: a cada estado do grafo corresponderá um estado na PLA; a cada sinal de entrada diferente encontrado no numerador da fração, sobre os arcos do grafo corresponderá uma entrada na PLA; a cada arco do grafo corresponderá um termo de produto na PLA. As equações da PLA são formadas observando-se que as setas que chegam nos estados "setam" os mesmos e as que deles partem os "resetam". A cada sinal de saída diferente encontrado no denominador das frações sobre os arcos do grafo corresponderá uma saída na PLA. As equações das saídas são obtidas como combinações dos estados e entradas da PLA.

As equações para a PLA não determinística da fig. 7 são as seguintes:

$RO = RO.\overline{LIBRX}.\overline{DIS} + EM.AC.P34;$

$EM = ENDOK + EM.\overline{P35}.\overline{DIS};$

$RP = EM.\overline{AC}.P34 + RP.\overline{LIBRX}.\overline{DIS};$

$MARCA = EM.AC.RO.P34 + EM.AC.RO.P35 + EM.\overline{AC}.P34;$

Para se chegar a estas equações algumas simplificações foram feitas, como, por exemplo, o estado RL não foi computado como estado, já que este estado está sempre ativo. As saídas RO e RP correspondentes a recepção ocupada e pacote rejeitado corresponderam aos próprios estados RO e RP, economizando portanto, 2 saídas da PLA. A transformação de autômato não determinístico em PLA é abordado com mais detalhes em {FLOY 80}.

As equações extraídas para o controlador corresponde a uma PLA com 13 entradas (6 entradas de sinal, 4 entradas de sinal complementadas, 3 estados), 4 saídas (1 estado, 1 saída, 2 estados/saídas) e 9 termos de produto. Portanto, obteve-se uma PLA 17 x 9 (Fig. 8).

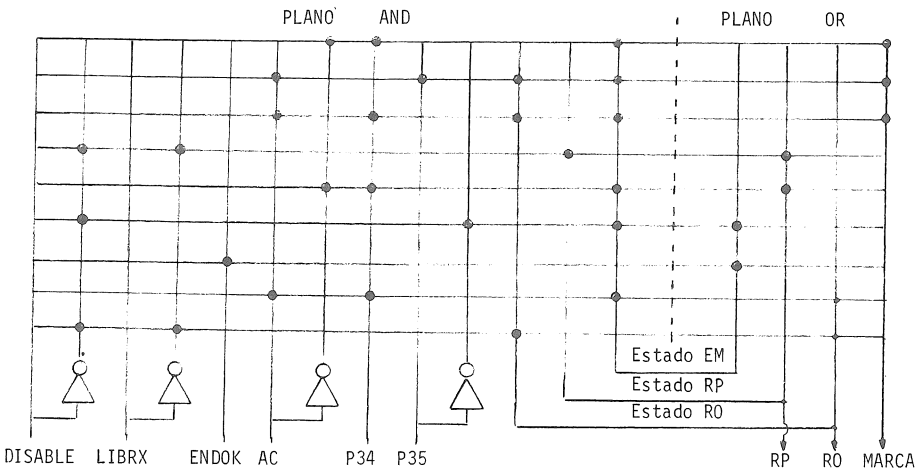


FIG. 8 - PLA DO CONTROLADOR DA FIG. 7

No exemplo do controlador pudemos observar os seguintes pontos:
TAMANHO - A PLA não determinística é 38% do tamanho da PLA determinística;
VELOCIDADE - A PLA não determinística é garantidamente mais rápida, pois, além de menor, apresenta menor número de programações nos planos AND e OR;
TEMPO DE PROJETO - A PLA não determinística é facilmente projetada, a partir do grafo de seu autômato, enquanto que a PLA determinística necessita de uma tabela de transição obtida a partir de seu autômato e um esforço de codificação de estados da forma mais conveniente, eliminação de estados redundantes e minimização das funções. Cabe observar que a minimização de funções só é viável de ser realizada manualmente para um número máximo de 6 variáveis de entrada (e estados). A partir daí pode ser realizada através de programas de minimização;
CLAREZA DO PROJETO - A PLA não determinística está claramente relacionada com a estrutura do problema, tornando mais fácil para o projetista extrair da máquina as informações necessárias (saída da máquina), bem como reconhecer, a cada instante, o estado em que a máquina realmente se encontra, apesar da máquina poder se encontrar em vários outros estados.

Faremos a seguir algumas observações que parecem explicar o fato da PLA não determinística ter apresentado um melhor resultado no exemplo do controlador.

Se analisarmos as duas PLA's, verificamos que a PLA não determinística, devido à não codificação de seus estados, não necessita dos mesmos complementados na entrada de seu plano AND, representando um ganho de área.

Outra observação a ser feita é que, como a máquina não determinística representa a forma direta com que o projetista vê sua máquina, geralmente seus estados são escolhidos de forma a representar eventos que devem ser gerados pela máquina, ou seja, suas saídas. Assim, os estados RO e RP da máquina não determinística apresentada representam duas das tres saídas da máquina, economizando espaço.

Outro fato importante é que a máquina não determinística já foi descrita por seu autômato na forma que representa o número mínimo de equações, dispensando, portanto, o trabalho árduo de minimizações.

O fato da PLA não determinística ter apresentado um número menor de termos de produto está relacionado ao fato de que esta máquina pode se encontrar em mais de um estado ao mesmo tempo. Quando,por exemplo,a máquina está no estado RO e vai para os estados RO e EM, não é necessário descrever uma equação que retire a máquina do estado RO, apenas é necessário equacionar a condição de ativação do estado EM. Na máquina determinística isto não ocorre, pois, sempre que a máquina passa para um novo estado,ela deve sair do estado anterior.

A referência {ULLM 84} enfoca o projeto de PLA's não determinísticas com codificação dos

estados não conflitantes, ou seja, não ativos ao mesmo tempo.

CONCLUSÃO

Apresentou-se neste trabalho a implementação de um controlador em PLA determinística e não determinística e esta última mostrou-se como a melhor solução. Embora neste exemplo isto pareça ser verdade, nem sempre a PLA não determinística é melhor que a determinística.

Vejamos como se comportam as duas PLA's. Se uma PLA não determinística tem N estados e I entradas, a PLA determinística teria, no pior caso, 2^n estados {HOPC 69}; Neste caso, com a codificação dos estados, a PLA determinística necessitaria de N bits para representá-los. Ou seja, o número de bits necessários para representar as duas máquinas seria o mesmo (é importante ressaltar que não estamos considerando a possibilidade de codificação dos estados da PLA não determinística {ULLM 84}. Entretanto, a PLA determinística necessitaria, no pior caso, de 2^{n+I} termos de produto para computar a função próximo estado. Neste pior caso, uma PLA determinística que implementasse esta máquina seria da ordem de $O(N)xO(2^n)$; esta PLA não poderia ser implementada na maioria dos casos, devido ao seu tamanho e velocidade.

Entretanto, em exemplos práticos, é mais comum para uma PLA não determinística ser convertida em uma PLA determinística com aproximadamente o mesmo número de estados (no exemplo do controlador a PLA determinística teria 8 estados, codificados em 3, e a não determinística 3 estados). Neste caso a PLA determinística seria da ordem de $O(\log n)xO(N)$, podendo representar uma implementação melhor que a PLA não determinística. Porém, outros fatores devem ser considerados. Vejamos o exemplo do controlador.

Para o controlador temos o seguinte quadro:

	PLA DETERMINÍSTICA	PLA NÃO DETERMINÍSTICA
Nº ENTRADAS	6	6
Nº SAÍDAS	3	1
Nº ESTADOS	3	3
Nº TERMOS PRODUTO	17	9

Portanto, para o controlador, o fato de termos aproveitado 2 estados como saídas e o fato de uma PLA não determinística não necessitar dos estados na forma não complementada reduziu a PLA em uma de suas dimensões. A outra dimensão da PLA é devida aos termos de produto, os quais foram menos numerosos na PLA não determinística.

BIBLIOGRAFIA

{BOOT 67} BOOTH, T.L. - Sequencial Machines and Automata Theory, Wiley, New York, 1967;

{FLOY 80} FLOYD, R.W. & ULLMAN, J.D. - The Compilation of Regular Expressions into Integrated Circuits, Report No. STAN-CS-80-798, Stanford University, April 1980;

{HOPC 69} HOPCROFT, J.E. & ULLMAN, J.D. - Formal Languages and their Relation to Automata, Addison Wesley, Reading, Mass., 1969;

{MEAD 80} MEAD, C. & CONWAY, L. - Introduction to VLSI Systems, Addison Wesley, Reading, Mass., 1980;

{OBRE 82} OBREBSKA, M. - Efficiency and Performance of Different Design Methodologies for Control Parts of Microprocessors, Microprocessing and Microprogramming, 10 (2, 3) , North-Holland Publishing Company, Amsterdam, 1982;

{RHYN 73} RHYNE, V.T. - Fundamentals of Digital Systems Design, Prentice-Hall, Inc., New Jersey, 1973;

{SILV 83} SILVA, H.T. & OLIVEIRA, C.E.T. - Circuito Integrado para Rede de Computadores, Ciência Hoje, Vol. 2, Nº 8;

{ULLM 84} ULLMAN, J.D. - Computational Aspects of VLSI, Computer Science Press, Inc., Maryland, 1984.

P. SALENBAUCH* e E.A. SCHMITZ**

SUMÁRIO

Este artigo examina as relações da área ocupada e tempo de operação para alguns tipos de somadores binários. Discute-se, entre outra, os tipos: "ripple-carry", "previsão de carry" e "previsão de carry logarítmico". Para este último apresenta-se um programa para gerar automaticamente seu lay-out.

ABSTRACT

This paper examines space and time bounds for some binary adders. It includes a discussion of: "ripple-carry", "carry-look-ahead" and "logarithmic look-ahead adders". A routine for automatic lay-out generation of logarithimic look-ahead adders is presented.

* Engenheiro eletrônico (ITA, 1969), mestrado em Engenharia de Sistemas e Computação, COPPE - UFRJ, 1972; Compiladores, Linguagens de Programação, Sistemas Operacionais;

** Engenheiro eletrônico (UFRGS, 1971), M.Sc. COPPE - UFRJ, 1973; Ph.D. Imperial College 1980; Projeto de Circuitos Integrados.

Endereço de ambos os autores: Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, CCMN, Cidade Universitária - Ilha do Fundão, Caixa Postal 2324 , CEP 20001, Rio de Janeiro, telefone (021) 290-3212, ramal 290.

O objetivo deste artigo é o estudo de alguns tipos somadores binários de N bits adequados para integração em grande escala. Ao estudar um determinado projeto de somador, levaremos em conta não somente o tempo de soma e o espaço ocupado pelo somador, como também a regularidade de cada módulo e o problema topológico de ligação entre estes diversos módulos.

Na seção 2, lembramos que teoricamente uma soma de operandos de qualquer comprimento pode ser feito em tempo constante, desde que tenhamos o circuito adequado - cujo espaço cresce exponencialmente com o número de bits. A seguir, focalizamos a situação inversa , em que mostramos que com um circuito fixo podemos somar operandos de qualquer comprimento com tempo crescendo linearmente com o número de bits dos operandos.

A seção 3 apresenta o "Ripple Carry", que é um dos esquemas mais simples de realizar a soma, tendo tanto o espaço como o tempo O(N). A seguir é mostrado o "Manchester Carry - Chain", um aperfeiçoamento do "Ripple-Carry", com algumas características próprias.

Na seção 4 é apresentado o método de "Carry Lookahead". A versão logarítmica do somador é discutida e um programa para geração é apresentado.

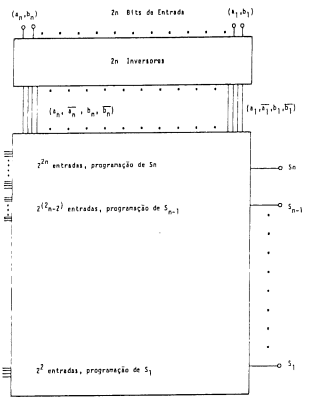
Finalmente, a seção 5 apresenta comparações entre os vários métodos e as conclusões finais.

COTA INFERIOR DE TEMPO E ESPAÇO

COTA INFERIOR DE TEMPO

Como sabemos, qualquer função lógica pode ser realizada em 3 etapas: na primeira, obtemos os inversos dos dados de entrada (se eventualmente estes já estão disponíveis, esta etapa pode ser dispensada); na segunda, obtemos os P-termos utilizando a operação lógica "E"; finalmente na terceira etapa obtemos a função desejada fazendo o "OU" dos P-termos obtidos na segunda etapa.

Como a soma binária de N bits nada mais é do que N funções de 2N, 2N-2, 2N-4, ..., 2 variáveis binárias, podemos utilizar esta idéia para obter um somador, como no esquema dado a seguir:



Esta é a maneira mais rápida de achar a soma, cujo tempo de operação é independente da largura dos operandos. Este tempo tem o valor da soma do tempo de um inversor mais o tempo do "E" mais o tempo do "OU". Infelizmente, no entanto, a área necessária é exponencial (basta ver que apenas para obtermos o s_n precisamos de (2 ** (2*N)) linhas).

O esquema acima apresentado supõe portas lógicas sem restrição de "fan-in". Se considerarmos que as portas podem ter "fan-in" máximo igual a 'R', então a cota inferior de tempo Winograd será:

TEMPO = O(LOG(R) (2+N))

Porém tendo uma função área:

A = O(2**N)

Brent mostrou ser possível construir somadores com "fan-in" limitado de forma que:

T = O((1+E) *LOG(R) N)

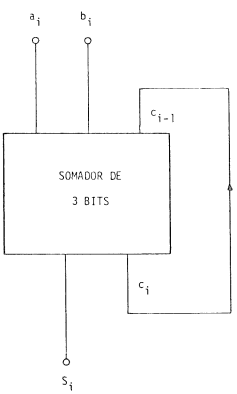
A = O(N*LOG N)

Onde E → Ø quando N → Infinito. (Brent)

COTA INFERIOR DE ESPAÇO

Tomando o extremo oposto do caso anterior, podemos também realizar a soma de N bits tendo apenas um somador de 3 bits (isto é, um circuito de complexidade constante). Com isto sacrificaremos o tempo de soma, obtendo somas em um tempo linearmente dependente da largura dos operandos.

Seria um esquema do seguinte tipo:



Este esquema, com apenas um somador, o carry de cada dígito é realimentado para realizar a soma dos bits de ordem imediatamente superior no ciclo seguinte da soma.

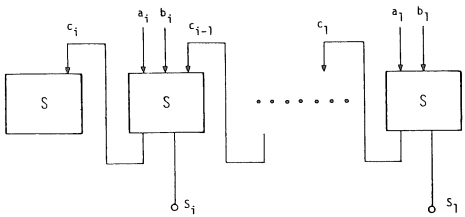
Neste caso temos: espaço O(N), tempo O(N) e espaço x tempo $O(N^2)$. Este esquema é o que provavelmente tem o melhor valor (espaço x tempo).

SOMADOR COM RIPPLE-CARRY

No caso simples a soma de N bits é dividida em N somadores de 3 bits, cada um calculando:

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

$$C_i = A_iB_i + (A_i + B_i).C_{i-1}$$



Donde:

ÁREA = O(N)

E:

TEMPO = O(3*N)

Pois em cada módulo acima, para se obter a soma a partir dos 3 sinais de entrada, necessitamos de 3 níveis lógicos.

MANCHESTER CARRY CHAIN

É uma variante do somador com ripple carry onde é colocado um meio mais rápido para propagação do carry. Ao invés do carry C_{i-1} ser processado como uma variável de entrada do bloco somador (gastando no mínimo 2 unidades de atraso para se propagar ao bloco seguinte) faz-se com que o carry seja gerado ou bloqueado em cada estágio por meio de chaves colocadas em série com o "percurso de carry". Esta operação de acionamento de chaves pode ser feita simultaneamente em todos os estágios de forma que o tempo de soma fica sendo dependente do comprimento físico da linha do carry.

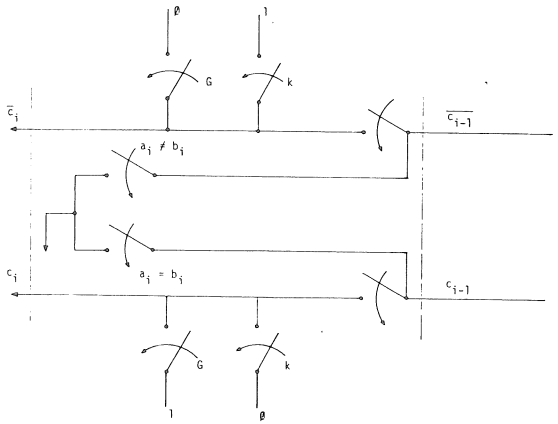
O diagrama a seguir ilustra a idéia.

Onde:

$P = A_i \oplus B_i$: Carry é propagado se $A_i \neq B_i$

$K = (-A_i).(-B_i)$: Carry de saída será zero ("Kill")

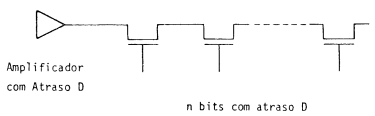
$G = A_i.B_i$: Carry de saída será um ("Gerador").



A área ocupada é O(N), porém temos dois casos a analisar para o tempo de propagação:
Caso 1 - Supondo que a propagação do carry seja em uma linha de transmissão com parâmetros distribuídos. No pior caso o tempo de propagação é A*L**2, onde L é o comprimento da linha.

T = O(L**2)

Caso 2 - Supondo que amplificadores sejam colocados a intervalos convenientes da linha de carry (Mead & Conway 80).



Temos (N/M) seções com atrazo 4*D. Escolhendo-se M de tal forma que o atrazo D seja igual ao de um nível de lógica:

TEMPO = (N/M)*4 = O(4*N/M)
COMO EM GERAL M = 4,
TEMPO = O(N)

SOMADOR COM PREVISÃO DE "CARRY"

PREVISÃO DE "CARRY"

Ao se observar o circuito de um somador notamos que o que determina basicamente o tempo

de soma é a espera de cada somador elementar pelo carry do estágio anterior.

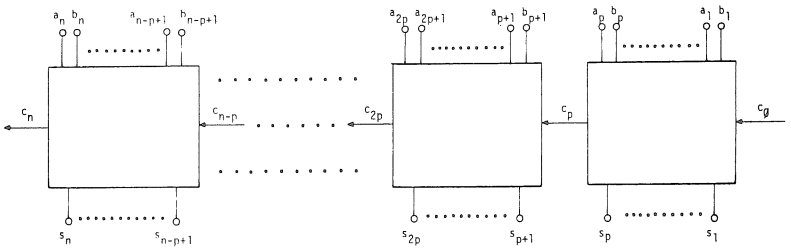
Uma idéia alternativa consiste em dividir os operandos de N bits, em grupos de P bits (vamos supor que N é múltiplo de P). Se pudéssemos saber instantaneamente qual seria o carry resultante de uma soma de um grupo de P bits, então seria possível saber-se o resultado da soma dos N bits com uma velocidade P vezes maior.

Na realidade, este tempo de previsão não é nulo. Vamos supor que o tempo de previsão de carry de um grupo de P bits é T. Neste caso o tempo de soma dos N bits será:

N/P*T * (Tempo de um Estágio)

Podemos notar que este esquema é realmente mais rápido, por um fator de P vezes, mas a sua velocidade continua sendo uma função linear de N.

Esta é a idéia central do somador de "Carry Lookahead", que tem um esquema do seguinte tipo:



Cada caixa acima tem um circuito convencional (ripple-carry) para obter a soma dos P dígitos e mais um circuito especial para obter o carry de saída (antes da soma).

Para este circuito teremos: espaço O(N), tempo O(N/P), espaço x tempo O(N**2/P).

Nos somadores de "Carry Lookahead" existem duas funções auxiliares de grande importância, que são definidos como:

$$G_i = A_i . B_i$$
$$P_i = A_i \oplus B_i$$

O G_i é a função de "Geração de Carry", e informa se no i-ésimo estágio será ou não gerado um carry. O P_i é a função de "Propagação de Carry", e informa se o i-ésimo estágio propagará ou não um carry que venha do estágio I-1.

Como sabemos que:

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$
$$C_i = A_i . B_i + A_i . C_{i-1} + B_i . C_{i-1}$$

Podemos deduzir que:

$$S_i = P_i \oplus C_{i-1}$$
$$C_i = G_i + P_i . C_{i-1}$$

A segunda expressão é bem intuitiva, informando que este estágio irá ter um carry na saí

da, se ele for gerado aqui ou propagado do estágio anterior.

A vantagem de usar estas funções auxiliares P e G é que teremos circuitos mais simples do que se calculássemos os carry's diretamente dos A_i e B_i. Assim, para um bloco de 4 bits (P = 4), teremos as seguintes funções para o carry de saída:

$$C_4 = G_4 + G_3.P_4 + G_2.P_4 + G_1.P_2.P_3.P_4 + C_\emptyset.P_1.P_2.P_3.P_4.$$

Esta fórmula é também bastante intuitiva, e significa que ou o carry é gerado no quarto estágio, ou então gerado no terceiro e propagado no quarto, e assim por diante.

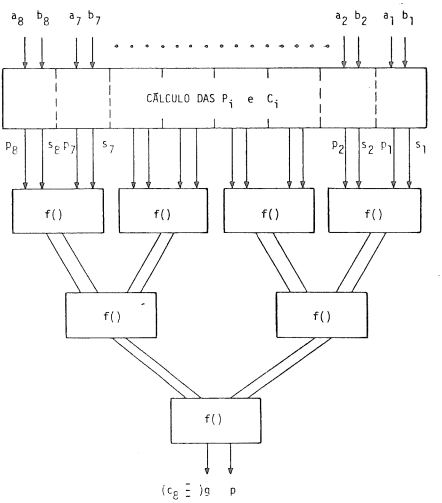
CARRY LOOKAHEAD LOGARÍTMICO

Uma outra solução, mais rápida e sofisticada, utiliza uma função auxiliar mais complexa:

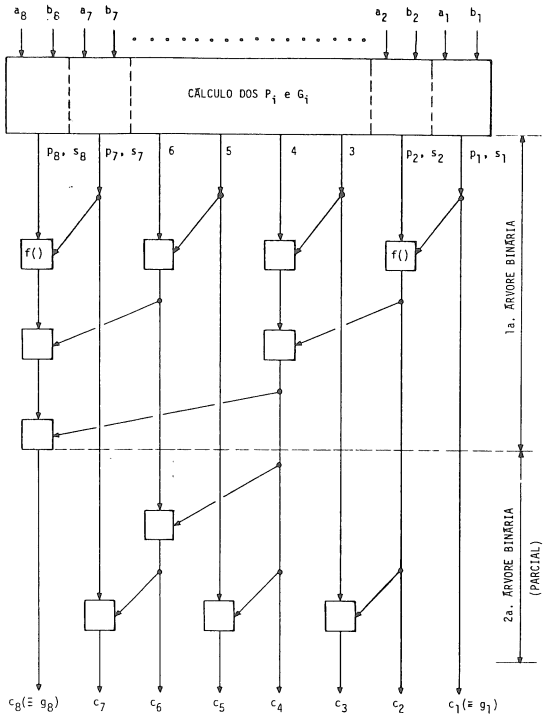
$$F((G_i, P_i), (G_j, P_j)) = (G_i + P_i.G_j, P_i.P_j).$$

Onde I e J são quaisquer (não necessariamente adjacentes). É possível mostrar que esta função pode ser usada para calcular todos os carry's e que ela é associativa.

Isto tem a vantagem de que estes cálculos podem ser iniciados em todos os estágios em paralelo, dando um circuito em forma de árvore binária:



Este circuito produz, em um tempo logarítmico, o carry resultante da soma dos N bits (no caso N = 8). No entanto, para obtermos os 8 bits da soma, não basta termos C_8. Necessitamos também ter C_1, C_2, ..., C_7. Para obtermos estes carry's restantes, precisamos de mais uma árvore binária (parcial), montado segundo o esquema a seguir:



Baseados nesta idéia, primeiramente descrita por Brent e Kung (79), podemos construir um somador, cujas características de tempo e espaço dadas por:

ESPAÇO = O(n.log(n))

TEMPO = O(log(n))

Para que o lay-out se torne regular Brent introduziu a noção dos "processadores brancos" e dos "processadores pretos" cujas funções lógicas são dadas por:

Processador preto: recebe como entrada - p_i, $\hat{p}_i$, g_i, $\hat{g}_i$

gerando: pout = $p_i p_j$

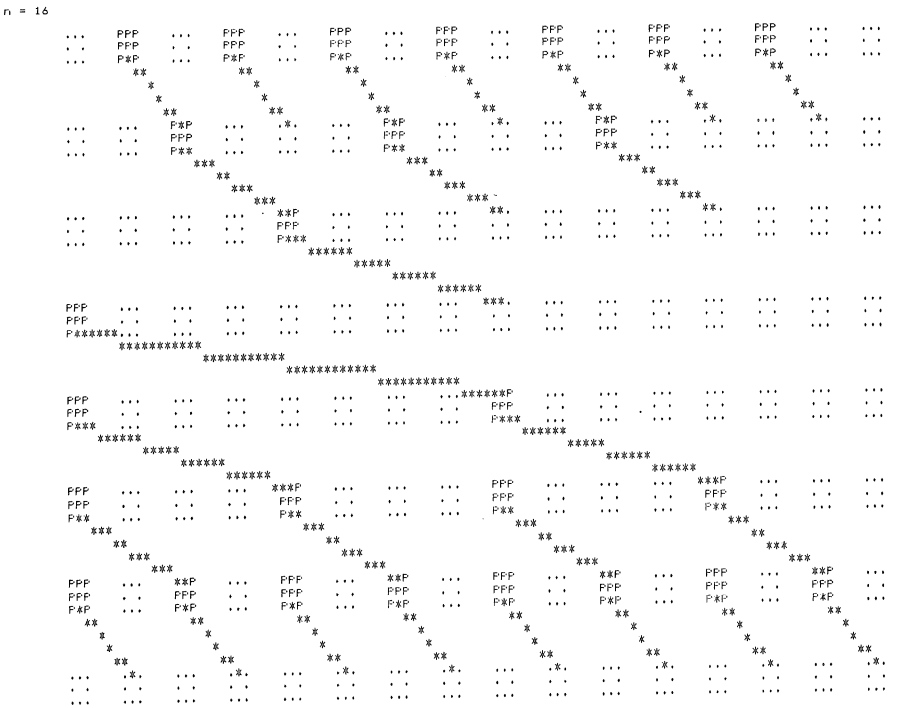
gout = $g_i + p_i g_j$

Processador branco: recebe como entrada (gin, pin) e gera dois grupos de sinais iguais tais que, gout = g_i e pout = p_i.

Finalmente, para a geração de soma basta acrescentar um estágio final que realize:

$S_i = P_i + C_{i-1}$.

Como o lay-out obtido é extremamente regular, temos no apêndice 1 o texto fonte de um programa para geração automática de somadores logarítmicos. Para adaptá-lo a uma dada tecnologia, basta se definir os lay-outs (em CIF, p.ex.) dos processadores brancos, pretos e de sua interconexão. A fig. abaixo mostra uma saída simbólica do programa para um somador de 16 bits.



CONCLUSÕES

Para somadores de tamanho moderado, a melhor opção continua sendo o Manchester Carry Chain, que tem tanto a área como o tempo linear com n.

Para somadores de tamanho grande, onde a velocidade é um fator importante, o circuito com previsão de carry logarítmico é o mais indicado, pois com ele temos um lay-out regular, adaptado para circuito com alta integração com uma penalidade de área que cresce

com n.log(n). A tabela abaixo mostra um resumo dos resultados obtidos.

SOMADOR	TEMPO	ESPAÇO
Ripple-Carry	O(3n)	O(n)
Manchester	O(n)	O(n)
Lookahead simples	$O(\frac{3}{4}\ n)$	O(n)
Lookahead logarítmico	O(2 log(n))	O(n.log(n))

BIBLIOGRAFIA

WINOGRAD, S. "On the time required to perform addition", J.A.C.M., V2-2, Abril 1965;

BRENT, R. "On the addition of binary members", Trans. on Computers, C-19, Agosto 1970;

MEAD C.A., Conway, L. "An introduction of VLSI systems", Addison-Wesley, 1980;

HWANG K. "Computer Arithmetic", John Wiley, 1979;

BRENT R., KUNG H. "A regular Lay-out for parallel adders", Technical Report, CMU-CS-79-131, Carnegie-Melon Univ., 1979.

```
/*
 *      Programa para gerar automaticamente
 *      o Layout do Somador Lookahead
 *
 *      Modo de usar: somador l
 *        onde l = log(2) n
 */

short   rx, ry;          /* Coordenadas */

short   l, n;            /* n = nr. de Bits, l = log(2) (n) */

main (argc, argv)
char **argv;
{
        register i;

        if ((l = atoi (argv[1])) <= 0) {        /* le o valor de l */
                printf ("l invalido\n");
                exit (1);
        }

        n = 1 << l;                             /* calcula n */

        for (i = 0; i <= l; i++) {              /* Linhas da Primeira Arvore */
                seek (0, i, 0);
                linha1 (1 << i);
        }

        for (i = 1; i < l; i++) {               /* Linhas da Segunda Arvore */
                seek (0, i+1, 0);
                linha2 (1 << (l-i) );
        }

        exit (0);

}       /* end main */

linha1 (i)
register i;
{
        register j;

        for (j = 0; j < n; j++) {
                if (ry == 0)
                        branco ();
                else if (j % i == 0) {
                        preto ();
                        fio (rx + (i >> 1), ry-1);
                } else
                        branco ();
                seek (1, 0, 1);
        }

}       /* end linha1 */

linha2 (i)
register i;
{
        register j;

        for (j = 0; j < n; j++) {
                if (j % i == (i >> 1) && j < n - i) {
                        preto ();
```

```
                        fio (rx+ (i>>1), ry-1);
                } else
                        branco ();

                seek (1, 0, 1);

        }

}       /* end linha2 */


fio (x, y)
{
        /*************************************************************
        *                                                            *
        *       Neste Ponto Insere-se o codigo para gerar uma        *
        *       ligacao do ponto (x,y) ao (rx,ry)                    *
        *                                                            *
        *************************************************************/

}       /* end fio */

seek (x, y, r)
{
        if (r) {
                rx += x;
                ry += y;
        } else {
                rx = x;
                ry = y;
        }

}       /* end seek */

branco ()
{
        /*************************************************************
        *                                                            *
        *       Neste ponto insere-se o Codigo para gerar            *
        *       um Processador Branco no ponto (rx,ry)               *
        *                                                            *
        *************************************************************/

}       /* end branco */

preto ()
{

        /*************************************************************
        *                                                            *
        *       Neste ponto insere-se o Codigo para gerar            *
        *       um Processador Preto no ponto (rx,ry)                *
        *                                                            *
        *************************************************************/

}       /* end Preto */

#SYS#
```

MODELAMENTO DA TENSÃO DE LIMIAR EM TRANSISTORES MOS GENERICAMENTE POLARIZADOS

Jader Alves de Lima Filho
ITAÚ COMPONENTES S.A.
Av. do Estado 5459
01515 São Paulo - SP

Trabalho realizado no Edinburgh Microfabrication Facility, Universidade de Edimburgo, sob o patrocínio FINEP/CNPq/Conselho Britânico.

RESUMO: Um preciso modelamento da tensão de limiar em transistores MOS é requerido ao se integrar subsistemas digitais em escala LSI e VLSI utilizando técnicas de projeto dinâmico.

Neste trabalho é apresentado um modelo teórico-experimental que descreve o comportamento da tensão de limiar em transistores MOS generícamente polarizados, considerando-se os efeitos devido a pequenas dimensões de canal. Incluindo flutuações devido à inerente não-uniformidade do processo de fabricação, um erro médio inferior a 10% foi obtido no confronto entre os valores calculados e experimentais da tensão de limiar, para uma larga faixa de dimensões de canal.

Embora os valores obtidos para os parâmetros de ajuste ("fitting parameters") estejam vinculados ao processo NMOS utilizado, pode-se recomendar uma aplicação genérica do presente modelo a transistores MOS, canal N ou P, com porta de silício policristalino ou de metal.

ABSTRACT: Both LSI and VLSI integration of digital circuits request an accurate modeling of MOSFET threshold voltage whenever dynamic techniques of designing are adopted.

The purpose of this work is to present a simplified theoretical-experimental model for the threshold voltage in generally-biased transistors, taking into account small-channel dimensions effects. Comparisons between calculated and measured values have shown an average error lower than 10% over a large range of channel dimensions.

Although the fitting parameters have values tied to the process of fabrication, one can recommend a general application of this model for both N and P channel MOS transistors.

GENERICAMENTE POLARIZADOS"

01. INTRODUÇÃO

Neste trabalho é apresentado um modelo teórico-experimental que descreve o comportamento da tensão de limiar em transistores MOS genericamente polarizados, considerando-se os efeitos devido a pequenas dimensões de canal.

A utilização de modelos que descrevem as características de transistores MOS através de expressões de ajuste torna-se importante em Projetos Auxiliados por Computador, permitindo que a simulação de circuitos em escala de integração LSI, ou mesmo VLSI, seja realizada com grande economia de tempo de CPU. Também, efeitos que requerem uma complexa análise teórica podem ser temporariamente modelados, com boa precisão, através deste método.

Os resultados experimentais são extraídos de transistores com diferentes dimensões de canal, fabricados segundo um processo NMOS, porta de Silício policristalino, disponível no Edinburgh Microfabrication Facility, utilizando-se, para tanto, um sistema automático de aquisição de dados Keitley/300, acoplado a um computador Digital PDP-11. Embora alguns valores apresentados estejam diretamente vinculados ao processo de fabricação utilizado, pode-se recomendar, de um modo geral, a aplicação do referido modelo a transistores MOS, com porta de Silício policristalino ou de metal, e com ajuste da tensão de limiar através de implantação. Para tanto, alguns dos parâmetros experimentais devem ser recalculados em função do novo processo de fabricação.

02. MODELAMENTO DA TENSÃO DE LIMIAR EM TRANSISTORES MOS

A tensão de limiar é analisada inicialmente sob condições de baixos campos, evoluindo para condição mais genéricas de polarização, levando-se em conta os efeitos decorrentes de pequenos dimensões de canal. Deste modo, torna-se também necessário uma análise do fator de corpo, e sua dependência com a polarização de substrato e dimensões de canal.

02.01. Tensão de limiar para VDS=0.1V e VS=0V

A tensão de limiar de um transistor MOS com fonte aterrada e VDS=0.1V é descrita por:

$$VTH(VSUB,Lef,Wef) = VTO(Lef,Wef) + \gamma(VSUB,Lef,Wef) \times$$

$$\times \left(\sqrt{|VSUB| + 2\emptyset FP} - \sqrt{2\emptyset FP} \right) \qquad (2.1)$$

onde VTO é a tensão de limiar do dispositivo sem polarização de substrato (VSUB=0), γ o fator de corpo |1,2,3,4| e Lef, Wef correspondem às dimensões efetivas de comprimento e largura de canal, respectivamente. Neste trabalho, a tensão de limiar, com VDS=0.1V, é determinada experimentalmente através de extrapolação linear na curva corrente de dreno x tensão de porta, utilizando-se o método dos mínimos quadrado |5,6|.

02.01.01. Fator de corpo

Fixando-se uma polarização de substrato, experimentalmente a dependência do fator de corpo com o inverso das dimensões de canal 1/Lef e 1/Wef, tem-se mostrado aproximadamente linear, como indicado nas figuras 2.1 e 2.2.

Assim, o fator de corpo pode ser escrito como

$$\gamma(VSUB,Lef,Wef) = \gamma o(VSUB) \left[1 - \frac{KL}{Lef} + \frac{KW}{Wef} \right] \qquad (2.2)$$

onde $KL = tg\alpha / \gamma L$ (ver figura 2.1)

$KW = tg\beta / \gamma W$ (ver figura 2.2)

e $\gamma o(VSUB)$ é o fator de corpo para um transistor de comprimento e largura grandes de canal, cuja dependência com VSUB está mostrada na figura 2.3. Os valores médios obtidos experimentalmente para KL e KW são 0.43 µm e 0.85µm, respectivamente.

A não-uniformidade do perfil de impurezas no substrato (perfil gaussiano após a implantação e recozimento) faz com que, para pequenos valores de |VSUB|, o fator de corpo assuma valores relativamente altos, uma vez que a largura da camada de carga espacial, sob a região de porta, é pequena. À medida em que |VSUB| aumenta, o mesmo apresenta um valor mais constante, aproximando-se de uma situação de dopagem uniforme no substrato.

No presente modelo, a dependência do fator de corpo com a polarização de substrato é descrita por:

$$\gamma o(VSUB) = (\gamma 1 - \gamma 2) \exp(-\delta |VSUB|) + \gamma 2 \qquad (2.3)$$

onde $\gamma 1 = \gamma o\ (VSUB=0)$

$\gamma 2 = (2\, \varepsilon o\, \varepsilon siqNA)^{1/2} / Cox$

δ ... parâmetro de ajuste

Para um transistor com Lef=27.5 μm e Wef=30.0 μm, variando-se |VSUB| de 0 a 5 volts, tem-se δ na faixa 0.27 - 0.30 V^{-1}, sendo 0.29V^{-1} o valor adotado. Quanto a γ_1 e γ_2 tem-se, respectivamente, 0.720 $V^{1/2}$ e 0.375 $V^{1/2}$

02.01.02. Tensão de limiar para VSUB=0 e VDS=0.1V

Fixando-se a polarização de substrato em zero volts, a dependência da tensão de limiar com o inverso das dimensões de canal também tem-se mostrado aproximadamente linear, como indicado nas figuras 2.4 e 2.5, justificando VTO ser descrito por:

$$VTO\ (Lef,Wef)\Big|_{VSUB=0} = VTHLD\Big|_{VSUB=0} \times \left[1 - \frac{\alpha L}{Lef} + \frac{\alpha W}{Wef}\right] \qquad (2.4)$$

onde VTHLD corresponde à tensão de limiar para um transistor com largura e comprimento grandes de canal, para VSUB = 0. Neste caso, VTHLD pode ser considerado como sendo a tensão de limiar de um capacitor MOS |1,2,7,8|. Os valores médios obtidos experimentalmente para αL e αW são , respectivamente, 0.22 μm e 0.68 μm.

A validade da expressão (2.1) foi comprovada experimentalmente utilizando-se transistores com diferentes dimensões de canal, conforme mostrado nas tabelas 2.1 e 2.2, sendo o erro médio obtido inferior a 10%, incluindo as flutuações devido à inerente não-uniformidade do processo de fabricação. O valor de VTHLD(VSUB=0) adotado é 0.66V, experimentalmente obtido em transistores com Wef=50μm e Lef=47.5μm.

VSUB (V)	VTH (EXP) (V)	VTH (CALC) (V)	ERRO %
0.0	0.608	0.545	10.4
-0.5	0.817	0.682	16.6
-1.0	0.892	0.775	13.1
-1.5	0.947	0.844	10.9
-2.0	0.982	0.892	8.58
-2.5	1.01	0.941	7.16
-3.0	1.04	0.978	5.72
-3.5	1.06	1.01	5.26
-4.0	1.08	1.04	4.32
-4.5	1.11	1.06	4.25
-5.0	1.12	1.08	3.23

TABELA 2.1 - Valores calculados e experimentais de VTH para Leff=1.5μm

VSUB (V)	VTH (EXP) (V)	VTH (CALC) (V)	ERRO %
0.0	0.623	0.653	-4.96
-0.5	0.825	0.842	-2.01
-1.0	0.914	0.970	-6.11
-1.5	1.00	1.06	-6.28
-2.0	1.08	1.14	-5.55
-2.5	1.14	1.20	-4.70
-3.0	1.20	1.25	-3.56
-3.5	1.26	1.29	-2.36
-4.0	1.31	1.33	-1.04
-4.5	1.37	1.36	0.94
-5.0	1.42	1.39	1.72

TABELA 2.2 - Valores calculados e experimentais de VTH para Leff=3.5μm

02.02.01. Tensão de limiar na Região Triodo

Considerando-se ainda VS=0, a presença de uma tensão de dreno VD causa um acréscimo na largura da região de carga espacial próximo ao dreno. Portanto, espera-se uma dependência da tensão de limiar com VD, acentuada à medida em que o comprimento de canal é diminuído. De modo a simplificar esta dependência assume-se, para dispositivos de canal curto, uma região depletada conforme mostrado na figura 2.6. Assim, a tensão de limiar para um transistor em sua região triodo, em função de VD, é dada por |9|.

$$VTH(VD,VSUB,Lef,Wef) = VTO(Lef,Wef) \Big|_{\substack{VSUB=0 \\ VD = 0.1V}} +$$

$$+ \; 1/2\,\gamma\,(VSUB,Lef,Wef) \; x \left\{ \sqrt{2\emptyset FP + |VSUB| + VD} + \sqrt{2\emptyset FP + |VSUB|} - 2\sqrt{2\emptyset FP} \right\} \quad (2.5)$$

onde γ (VSUB,Lef,Wef) e VTO (Lef,Wef) são dados por (2.2) e (2.4), respectivamente.

Assumindo agora VS > 0 e VD=0, analogamente ao caso anterior, a largura da região depletada é alterada junto à fonte. A configuração da região de carga espacial adotada é mostrada na figura 2.7. Combinando-se ambos os casos, i.e., VS $\neq$ 0 e VD $\neq$0, o potencial de superfície é dado por:

$$\emptyset s(x) = \frac{VD}{Lef}\, x + \frac{VS}{Lef}\, (Lef-x) + 2\emptyset FP + |VSUB| \qquad (2.6)$$

e, admitindo-se junção N+/P abrupta e concentração uniforme de impurezas (NA) no substrato, a largura da região depletada é descrita por |1,2,8|

$$xd(x) = \left\{ \frac{2\,\varepsilon o\,\varepsilon si\,\emptyset s(x)}{qNA} \right\}^{1/2} =$$

$$= \left\{ 2\,\varepsilon o\,\varepsilon si \left[\frac{VD}{Lef}\, x + \frac{VS}{Lef}\, (Lef-x) + 2\emptyset FP + |VSUB| \right] \right\}^{1/2} \qquad (2.7)$$

A carga espacial, sob o eletrodo de porta, é dada por:

$$QB^* = q.NA.Wef \int_0^{Lef} xd(x)dx \qquad (2.8)$$

Substituíndo (2.7) em (2.8) e integrando, tem-se

$$QB^* = Lef.Wef \sqrt{2\varepsilon_o \varepsilon_{si} qNA}\, x \frac{2}{3(VD-VS)} \; x$$

$$x \left[(VD+|VSUB|+2\emptyset FP)^{3/2} - (VS+|VSUB|+2\emptyset FP)^{3/2} \right] \qquad (2.9)$$

A tensão de limiar em uma estrutura MOS é definida por |1,2,7,9|:

$$VTH = VFB + 2\emptyset FP + \frac{QB^*}{Cox.Wef.Lef} \qquad (2.10)$$

onde VFB corresponde à tensão de banda plana. Substituíndo (2.9) em (2.10) e reagrupando termos em função de VTO e γ , a tensão de limiar para um transistor MOS, genericamente polarizado em sua região triodo, é dada por:

$$VTH(VS,VD,VSUB,Lef,Wef)=VTO(Lef,Wef)+\gamma(VSUB,Lef,Wef)\; x$$

$$x \left\{ \frac{2}{3(VD-VS)} \left[(VD+|VSUB|+2\emptyset FP)^{3/2} - (VS+|VSUB|+2\emptyset FP)^{3/2} \right] \right.$$

$$\left. - (2\emptyset FP)^{1/2} \right\} \qquad (2.11)$$

02.02.02. Tensão de limiar na saturação

Tem-se observado que, transistores na saturação apresentam um decréscimo em sua tensão de limiar quando VD tem o seu valor aumentado, predominantemente em dispositivos de canal curto. No presente modelo este efeito é descrito por:

$$VTSAT(VS,VD,VSUB,Lef,Wef)=VTH(VS,VD,VSUB,Lef,Wef) -$$

$$- \alpha(Lef)\,(VD-VDSAT) \qquad (2.12)$$

onde VTSAT corresponde à tensão de limiar para um transistor saturado, VDSAT à tensão de saturação do dispositivo, VTH à tensão de limiar na região triodo, dada por (2.5) ou (2.11) e α(Lef) um parâmetro de ajuste, dependente do comprimento de canal.

Na figura 2.8 tem-se a variação da tensão de limiar na saturação com a tensão de dreno, para transistores com diferentes comprimentos de canal (Lef=0.5μm, 1,5μm e 3,5μm). No caso, VGS é fixado em 200mV acima da tensão de limiar, de modo que VDX=VD-VDSAT $\simeq$ VD com boa aproximação. O valor da tensão de limiar foi obtido através de extrapolação linear, no ponto de máxima derivada, pelo método dos mínimos quadrados, na curva $\sqrt{IDS}$ x VGS.

A dependência do parâmetro α com o comprimento de canal é mostrado na figura 2.9, adotando-se, portanto, uma variação linear:

$$\alpha\,(Lef) = \frac{hl}{Lef} \qquad (2.13)$$

sendo 4.28×10^{-2} um o valor médio encontrado para o parâmetro de ajuste hl.

Os resultados teóricos e experimentais da tensão de limiar para transistores genericamente polarizados, podem ser comparados através das tabelas 2.3 e 2.4. Os valores teóricos são calculados através da expressão (2.12), com o valores anteriormente especificados dos parâmetros de ajuste, e fixando-se VSUB=-2.5V.

VD (V)	VS (V)	VTSAT (EXP)	VTSAT (CALC)	ERRO (%)
2.00	0.95	1.05	1.08	-2.86
3.00	1.88	1.12	1.14	-1.79
4.00	2.85	1.15	1.20	-4.35
5.00	3.79	1.21	1.24	-2.48
6.00	4.77	1.23	1.29	-4.88
7.00	5.74	1.26	1.32	-4.76
8.00	6.71	1.29	1.36	-5.43

TABELA 2.3 - Valores de VTSAT (medidos e calculados) para um transistor com Wef = 30μm e Lef = 1.5μm

VD (V)	VS (V)	VTSAT (EXP)	VTSAT (CALC)	ERRO (%)
2.00	0.74	1.26	1.40	-11.11
3.00	1.68	1.32	1.51	-14.39
4.00	2.60	1.40	1.61	-15.00
5.00	3.49	1.51	1.69	-11.92
6.00	4.38	1.62	1.77	- 9.26
7.00	5.27	1.73	1.85	- 6.94
8.00	6.15	1.85	1.92	- 3.78

TABELA 2.4 - Valores de VTSAT (medidos e calculados) para um transistor com Wef = 6.0μm e Lef = 3.5μm

O conhecimento do decréscimo da tensão de limiar com a tensão de dreno, para um transistor saturado, torna-se importante ao se utilizar transistores de canal curto como chave. Variações na tensão de dreno de um transistor, inicialmente em seu estado de corte e tendo flutuante sua porta, pode levar o dispositivo a uma subcondução, ou mesmo a uma condução.

03. CONCLUSÃO

Procurou-se estabelecer, neste trabalho, um simplificado modelo teórico-experimental, baseado em parâmetros de ajuste ("fitting parameters") que descreva, com razoável precisão, o comportamento da tensão de limiar em transistores MOS, e que possa bem adaptar-se aos programas de auxílio a projetos, como, por exemplo, SPICE |10| e MSINC |11|.

Para tanto, foram definidas expressões para o cálculo da tensão de limiar em transistores MOS, sob qualquer condição de polarização, levando-se em conta sua dependência com a largura e comprimento efetivos de canal. Incluindo flutuações devido à inerente não-uniformidade do processo de fabricação, obteve-se um erro medio não superior a 10% no confronto entre os valores calculados e experimentais da tensão de limiar, para uma larga faixa de dimensões de canal ($0.5\mu m \leq Lef \leq 47.5\mu m$; $6.0\mu m \leq Wef \leq 50.0\mu m$) e de polarização ($0 \leq |VSUB| \leq 5V$; $0 \leq VGS \leq 12V$, $0 \leq VDS \leq 12V$).

Embora os valores obtidos para os parâmetros de ajuste estejam vinculados ao processo NMOS utilizado, pode-se recomendar uma aplicação genérica do presente modelo a transistores MOS, canal N ou P, com porta de silício policristalino ou de metal.

04. BIBLIOGRAFIA

01. A.S.Grove - "Physics and Technology of Semiconductor Devices", John Wiley and Sons, 1967

02. S.M.Sze - "Physics of Semiconductor Devices", John Wiley and Sons, 1981

03. R. Crawford - "MOSFET in Circuit Design", Texas Instruments Serie, McGraw Hill, 1967

04. J.A. Lima - "Projeto e Elaboração de uma Unidade Lógica e Artmética Utilizando Tecnologia NMOS" - Tese de Mestrado apresentada à EPUSP, 1980

05. A. Ralston - "A First Course in Numerical Analysis", McGraw Hill, 1965

06. M. Spiegel - "Estatística" - Coleção Schaum, McGraw Hill do Brasil, 1974

07. R. Cobbold - "Theory and Application of Field - Effect Transistors", John Wiley and Sons, 1970

08. C. Sah - "Characteristics of the Metal-Oxide-Semiconductor Transistor" - IEEE Transactions on Electron Devices, July 1964

09. E. Sun and J. Moll - "A Simple Analytical Short Channel MOS Model" - Technical Report, Hewlett Packard Laboratories, 1978

10. L.W. Nagel - "SPICE2: A Computer Program to Simulate Semiconductor Circuits" - Memo nº ERL-M-520 - 09.05.75, University of California

11. "MSINC - A Modular Simulador of Non-Linear Electronic Circuits" - A4 Version User's Guide, Stanford University, 1976

JALF
15.04.85

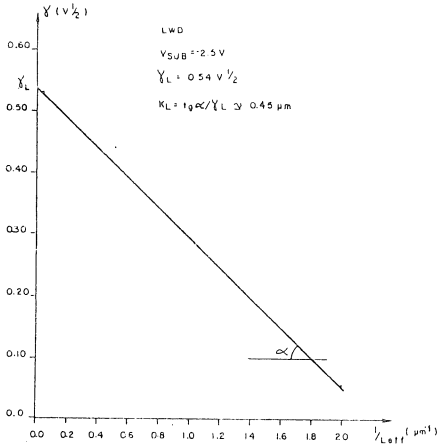


FIGURA 2.1 - Dependência do fator de corpo com o comprimento de canal

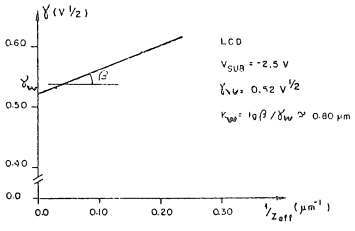


FIGURA 2.2 - Dependência do fator de corpo com a largura de canal

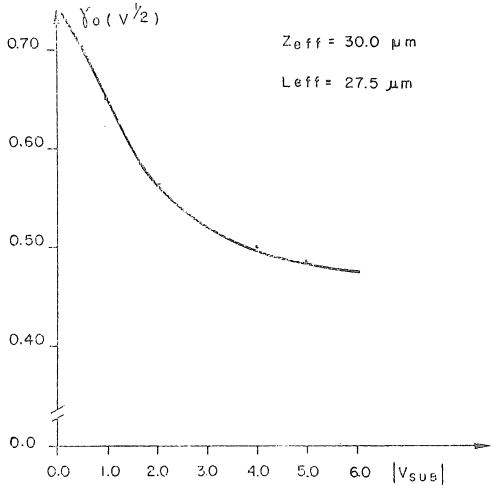


FIGURA 2.3 - Dependência do fator de corpo com VSUB para o referido processo de fabricação

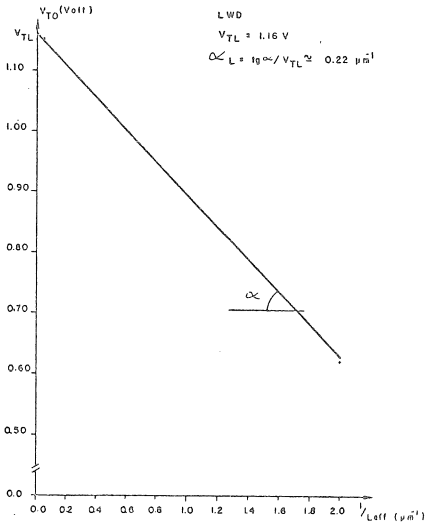


FIGURA 2.4 - Dependência de VTO com o comprimento de canal

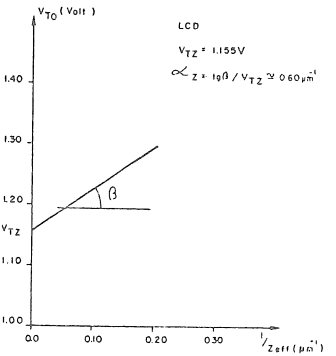


FIGURA 2.5 - Dependência de VTO com a larugura de canal

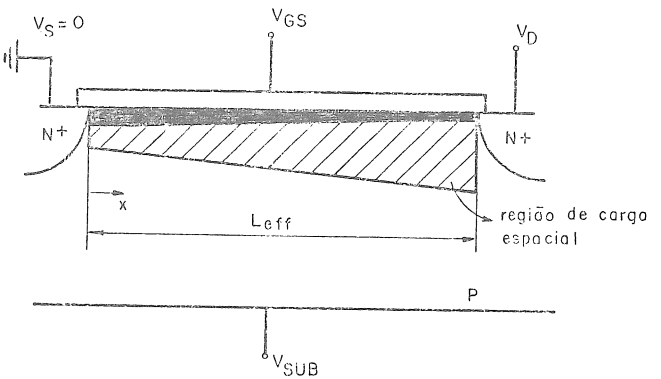


FIGURA 2.6 - Região de Carga espacial adotada para VD>0 e VS=0

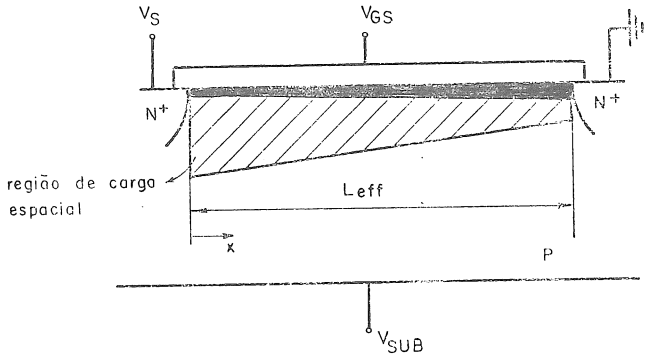


FIGURA 2.7 - Região de carga espacial adotada para VS>0 e VD=0

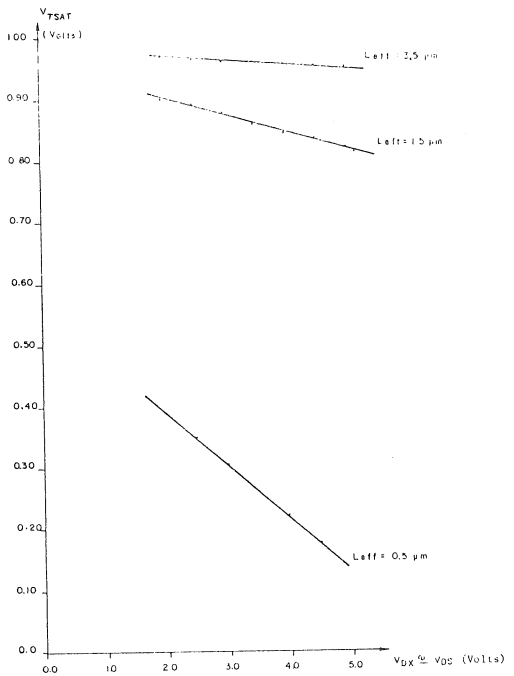


FIGURA 2.8 - Decréscimo da tensão de limiar com a tensão de dreno, na saturação

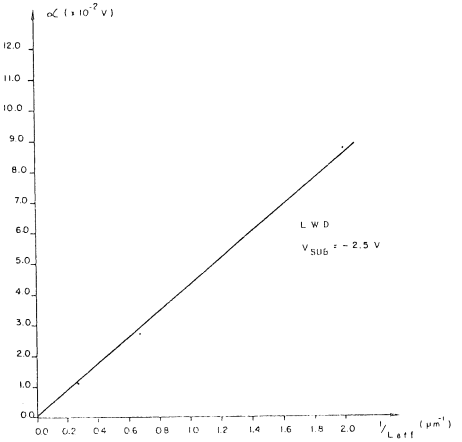


FIGURA 2.9 - Comportamento do parâmetro α com o comprimento de canal

PROJETO DE UM CIRCUITO INTEGRADO DEDICADO PARA APLICAÇÃO EM EQUIPAMENTO DE INTERPOLAÇÃO DE VOZ

A.S.Pires*, D.J.S.Conti**, H.J.Malavazi F.***

RESUMO

É apresentado o projeto de um circuito integrado digital dedicado elaborado para aplicação no equipamento MCP-60A em desenvolvimento no CPqD/TELEBRÁS. São enfocados principalmente a metodologia de concepção, os problemas e soluções encontrados ao longo do projeto e a filosofia de testabilidade adotada.

ABSTRACT

The design of a custom digital integrated circuit prepared for aplication on the MCP-60A equipment being developed on CPqD/TELEBRÁS is presented. The main topics focused are the design methodology, the problems and solutions found through all the phases of the design and the testability philosophy adopted.

* Eng.Eletrônico (UFRJ, 1983); Projeto de circuitos integrados, CPqD - TELEBRÁS.

** Eng. Eletrônico (UNICAMP, 1984); Projeto de equipamentos eletrônicos digitais, FTPT-TELEBRÁS

*** Eng.Eletrônico (UNICAMP, 1974); Projeto de circuitos integrados, CPqD-TELEBRÁS.

Rodovia Campinas-Mogi Mirim km 118,5
Caixa Postal 1579 - Telefone 59.1476

1. INTRODUÇÃO

É apresentado um circuito integrado digital, denominado TC8, componente de uma série de cinco dispositivos, ora em desenvolvimento no CPqD/TELEBRÁS, que visam atender ao equipamento de Interpolação Digital de voz denominado MCP-60A.

A opção de utilizar circuitos dedicados para o equipamento MCP-60A, traz como consequência uma série de vantagens, objeto de uma discussão posterior.

Para compreender o funcionamento do dispositivo TC8, torna-se necessário localizá-lo no contexto do MCP-60A, o que é feito a seguir.

Em equipamentos de interpolação de voz, de uma maneira geral, o processo de interpolação está fundamentado na otimização da utilização de seus canais, ou seja, considerando um canal genérico, pode ser demonstrado que a porcentagem de tempo que este canal contém informação útil de voz em relação ao período em que está sendo ocupado é da ordem de 25%. O processo de interpolação de voz consiste, portanto, em utilizar a "ociosidade" deste canal para enviar informações de outros canais de voz.

É necessário então detectar a presença de voz para cada canal do equipamento, ou seja, determinar se o canal está ativo ou não. Adicionalmente deve ser detectada a presença de eco nestes canais para que seja eliminado, uma vez que não representa informação útil, embora seja voz.

Para determinar a atividade dos canais é utilizado um circuito denominado Detector de Voz, e para o eco é utilizado um circuito denominado Supressor de Eco.

A aplicação do circuito integrado TC8 no equipamento MCP-60A é a de justamente auxiliar na realização destas duas funções (Detecção de Voz e Supressão de Eco).

Para tanto, o TC8 está estruturado funcionalmente de maneira a atender estes dois pontos de aplicação. Tal fato lhe confere um caracter particular de possuir 3 circuitos independentes, quais sejam : um Filtro passa baixas de 1a. ordem, um contador endereçável ("Hangover") e um Detector de faixa de frequências, este último podendo atuar também como "Hangover" a partir de programação externa.

Através de programação externa ("estrapes") configuram-se também alguns parâmetros que alteram as características destes 3 circuitos, conferindo ao dispositivo uma versatilidade adicional no tocante à sua utilização.

Para o caso do Filtro Digital tem-se a opção de frequências de corte distintas, para o caso do contador endereçável ("Hangover"), a opção do limite de contagem, e para o caso do Detector de faixa de frequências tem-se a opção sobre a faixa desejada, salientando-se ainda que todas as opções estão restritas ao número de terminais de programação reservados.

A figura 1.1 mostra o diagrama em blocos simplificado onde, por questões de simplicidade, foi omitido o bloco responsável pelas ondas de controle internas ao disposi

tivo. Deve ser notado ainda que a utilização de Memórias RAM nestes circuitos possibilita a operação em regime de partilhamento no tempo ("time-sharing"), pois a cada endereço da memória corresponde um dado canal, resultando num processamento único para todos os canais.

A estratégia adotada de não restringir o TC8 a um único ponto de aplicação no equipamento MCP-60A, o que ocorre também com outro dispositivo componente deste elenco de cinco circuitos integrados dedicados, reflete diretamente uma redução substancial de componentes que, aliada à complexidade destes circuitos e sua demanda estimada em pelo menos 4000 unidades, justifica o enfoque de projeto dedicado ou "full-custom".

O dispositivo TC8 caracteriza-se como um circuito integrado digital VLSI que utiliza tecnologia CMOS de 3 micra, de porta de silício.

Contém aproximadamente 8300 transistores, dos quais 5400 juntamente com 3500 capacitores formam 1664 bits de memória RAM, e 2900 compõem o restante do circuito.

Para se ter uma idéia da redução de componentes causada pela sua utilização, pode-se dizer que o TC8 contém o equivalente a 50 CI's TTL e 7 CMOS, sendo 23 MSI, o que representa em última análise, uma redução de aproximadamente 2 placas em cada uma das aplicações do dispositivo no equipamento.

2. PROJETO LÓGICO E ELÉTRICO

Um componente importante desse dispositivo é a memória, que teve rígidas especificações de funcionamento para que atendesse às necessidades dos circuitos. Como ela seria utilizada em quatro dispositivos, foi feito um projeto único de memória que poderia ser utilizada em todos os dispositivos. Essa memória é objeto de um trabalho específico e portanto aqui serão apresentadas apenas suas características principais : $t_{leitura} = 250ns$, $t_{escrita} = 390ns$.

Para o restante do circuito foi utilizada uma metodologia "top-down". Os blocos funcionais principais foram simulados logicamente usando a biblioteca de células padrão. Depois foi feito o projeto elétrico e o lay-out de todas as células visando atender às especificações de tempo de propagação e de mínima área do dispositivo. Procurou-se fazer, sempre que possível, para cada célula um único projeto que pudesse ser utilizado em todos os pontos de aplicação dessa célula no dispositivo visando agilizar o projeto e facilitar a concepção da planta baixa ("floor-planning").

Nas simulações foi utilizada uma margem de segurança da seguinte forma :

- As simulações lógicas foram feitas utilizando-se uma frequência de operação 25% maior que a nominal.
- As simulações elétricas utilizaram parâmetros de processo de pior caso de velocidade (mínimo fornecimento de corrente pelos transistores), temperatura de 80ºC e alimentação de 4,6V.

As cargas devidas às trilhas que percorrem o dispositivo foram estimadas a partir da

concepção da primeira planta baixa.

Dentre as várias células projetadas serão destacadas no presente trabalho as seguintes : SOMADOR COMPLETO, COMPARADOR, "BUFFERS" DE RELÓGIO, FLIP-FLOP TIPO D DINÂMICO e INTERFACES TTL DE ENTRADA E SAÍDA sobre cujos projetos serão tecidos alguns comentários a seguir.

O dispositivo apresenta somadores de módulos 4, 6 e 14. Dois tipos de implementação foram analisados e comparados : os de transportes série e paralelo. Optou-se pelo transporte série por este possuir as vantagens de ocupar menos área e não apresentar um aumento do número de transistores nas portas básicas (NANDs e NORs com múltiplas entradas) quando cresce o número de bits da soma. Além disso, o seu tempo de propagação (mais lento) demonstrou-se suficientemente bom para os casos simulados. Outra vantagem é a possibilidade de se usar uma célula básica de soma de um bit que precisa apenas ser encadeada n vezes quando se necessita somar n bits.

Essa argumentação também definiu o tipo de comparador adotado cuja célula básica é mostrada na figura 2.1.

A célula de FLIP-FLOP tipo D tem uma importância muito grande para o projeto em questão posto que ela, além de ser usada perto de 70 vezes, situa-se nos caminhos críticos do circuito em termos de tempo de propagação. Foi feita a escolha de um flip-flop dinâmico pelas vantagens de maior velocidade e menor área ocupada. A desvantagem principal deste flip-flop que é a perda de carga no caso de períodos grandes de relógio, não oferecia problemas posto que a frequência nominal do dispositivo era de 2,048 MHz.

O projeto dos "buffers" de relógio foi feito visando conseguir que este sinal chegasse a todos os pontos do dispositivo com o mínimo de atraso e com as duas fases atingindo a tensão de limiar de condução (tensão de "threshold") no mesmo instante, o que é necessário para o bom funcionamento do flip-flop. Isto foi conseguido escolhendo-se um tamanho adequado para as dimensões dos transistores dos "buffers".

É apresentada uma tabela, Fig. 2.2, com os tempos de propagação de cada célula aqui tratada, bem como as respectivas cargas utilizadas na simulação elétrica.

3. TESTABILIDADE

Um ponto importante que deve ser considerado já na fase do projeto lógico é o relativo à testabilidade do circuito integrado, pois é nesta fase que eventuais alterações do circuito, visando uma estrutura que facilite seu teste, podem ser mais facilmente incorporadas ao projeto.

Analisando o diagrama da figura 1.1, nota-se claramente a estrutura realimentada pertinente aos 3 blocos principais do dispositivo, além do mais observa-se o número reduzido de saídas, o que prejudica a observabilidade dos nós internos na fase de testes.

Adicionalmente, a presença de memória dificulta o teste, uma vez que ela deve ser

testada integralmente e se encontra no elo de realimentação; este último fato contribuindo negativamente para a monitoração de um dado canal, uma vez que o período de repetição das informações deste canal é o período necessário para percorrer todos os 64 endereços da memória, alongando desta forma o padrão de teste.

A estratégia adotada para o caso do TC8 consiste em dividir o padrão de teste em 2 partes; uma correspondente ao teste da memória de forma isolada, e outra correspondente ao teste dos demais circuitos.

Para o teste da memória é realizada a escrita e leitura nos 64 endereços, onde o conteúdo de uma posição é o valor do seu próprio endereço ou seu complemento, dependendo da fase do teste.

Para o teste do restante dos circuitos, os controles da memória são configurados de forma a variar apenas o bit menos significativo de endereço. Esta providência é utilizada para realizar testes distintos para cada um dos dois endereços fixados, ou seja :

Para o caso do Filtro Digital, o conteúdo de cada endereço simula uma resposta particular no tocante à frequência de corte e a uma dada condição de resposta ao degrau.

Para o caso do Contador endereçável o conteúdo de cada endereço simula uma condição distinta para um dado limite de contagem.

Para o caso do Detector de faixa de frequências o conteúdo de cada endereço representa uma condição distinta para uma dada faixa de detecção ou mesmo um certo limite de contagem, conforme a configuração em que se encontre (Detector ou "Hangover").

Deve ser observado que existe a necessidade de circuitos adicionais para implementar estas funções; com relação ao teste da memória deve-se "quebrar" a estrutura realimentada existente para se realizar a escrita e leitura dos dados desejados e para o teste dos demais circuitos deve-se alterar alguns sinais de controle da memória.

Fica patente portanto a necessidade de se efetuar um estudo sob o enfoque da testabilidade do dispositivo já na fase de projeto. A solução nunca é única e deve ser obtida levando-se em conta o compromisso existente entre a complexidade adicional em termos de circuitos e a economia ocasionada em termos da profundidade do padrão de teste e do grau de cobertura atingido.

Como exemplo específico, a metodologia adotada resultou numa redução de um padrão inicial de aproximadamente 25 Kbits para o padrão atual que contém em torno de 2Kbits, sendo para tanto necessária a utilização de 3 terminais para teste e 252 transistores adicionais (51 mux 2:1, 1 flip-flop, 9 inversores, 5 portas).

4. PLANTA BAIXA E LAYOUT FINAL

A primeira planta baixa do circuito foi feita no início da fase de simulações elétricas com o objetivo de se obter uma primeira avaliação dos comprimentos das trilhas

de sinal. Essa avaliação era importante para o dimensionamento das cargas máximas das células que estavam sendo projetadas.

Nessa primeira planta baixa preocupou-se apenas com a disposição relativa das células e suas interligações. Não ficaram definidas as posições das trilhas de alimentação e no caso das trilhas de sinal não se fez diferenciação entre interligação em polisilício e em metal. O tamanho das células foi baseado em estatísticas de número de dispositivos por área de silício extraídas de células projetadas anteriormente.

O principal problema encontrado na disposição das células foi o fato de existirem muitos barramentos de sinal no interior dos blocos principais do circuito : 14 bits no filtro digital, 8 bits no detector de faixa de frequências e 4 bits no "hangover" fixo. Para evitar os problemas de roteamento e espaço ocupado por esses barramentos, optou-se por uma estrutura em que todas as células têm a mesma altura (altura essa que é definida pelo tamanho de um bit da memória RAM) e incluem passagens de sinal em seu layout. Assim, o sinal vindo da saída da memória, entra na 1a. célula, cuja saída entra na próxima célula e/ou passa através dela para entrar numa terceira. Depois o sinal volta através de todas as células para entrar na memória novamente (Fig. 4.1).

Para conseguir um maior grau de compactação, foram definidas rígidas especificações de layout para as células que se encaixariam na estrutura descrita. Essas especificações foram por exemplo :

a) A comunicação entre células do mesmo tipo que se encontravam em fila teria que se fazer diretamente sem que a interligação ocupasse espaço entre as células, ou seja, no caso de comparadores e somadores a saída de "carry" de uma célula teria que se ligar diretamente com a entrada de "carry" da próxima célula. (Fig. 4.2).

b) Os sinais usados em células que estão em fila, tais como CLOCK, $\overline{\text{CLOCK}}$ e CLEAR nos FLIP-FLOPS e sinais de controle dos multiplexadores teriam que passar através das células e também se ligar diretamente de uma célula para outra quando as células se justapusessem . (Fig. 4.3.)

Depois de terminados o projeto elétrico e o layout das células, foi preparada uma planta baixa definitiva já levando em conta todos os detalhes de layout tais como trilhas de metal e polisilício, trilhas de alimentação, posição dos contatos, posição dos PADs, etc.

O layout final do circuito foi editado no computador gráfico e tem uma área aproximada de 4,2mm x 3,4mm.

5. BIBLIOGRAFIA

- Especificação do Circuito Integrado Filtro Digital e Hangover (TC8) - H.J.Malavazi, Dante J.S.Conti, R.Marche - CPqD/TELEBRÁS - abril/84.
- Projeto MCP-60A, Documento de Objetivos e Requisitos - CPqD/TELEBRÁS - março/84.
- A Digital Multifrequency Detector for PCM Systems - J.M.Laraia, A.L.P.Janzon, L. O.F.Gonçalves - Custom Integrated Circuits Conference - 1985.
- RAM Dinâmica Modular - J.M.Laraia, A.L.P.Janzon - V Simpósio Brasileiro de Microeletrônica - 1985

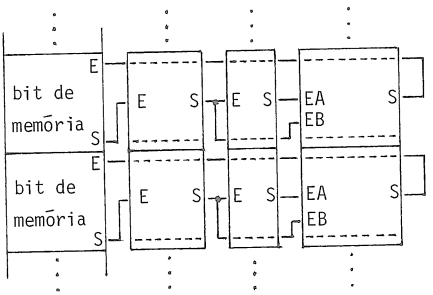


Figura 4.1

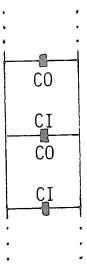


Figura 4.2

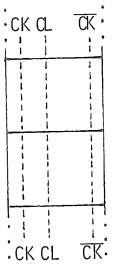


Figura 4.3

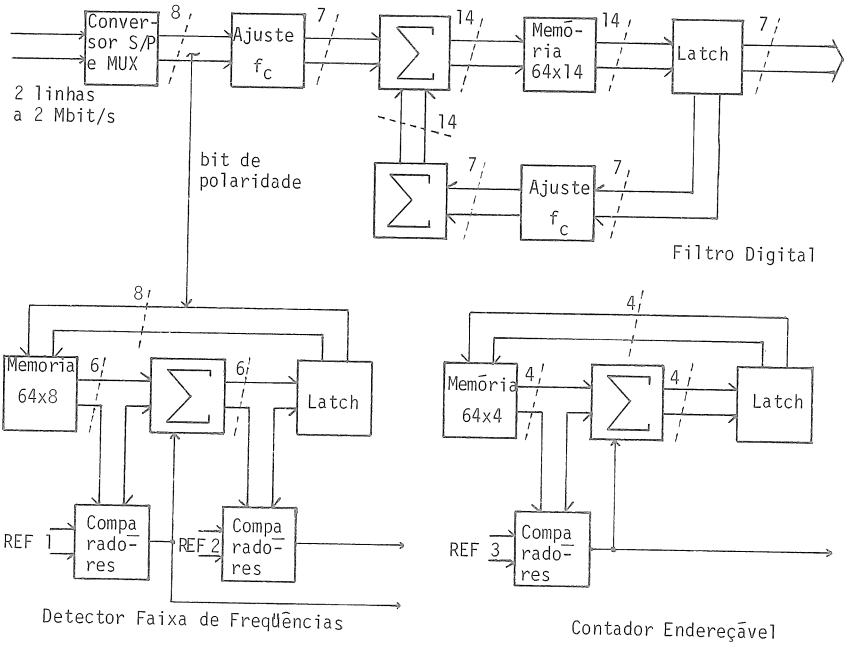


Figura 1.1

CÉLULA		CARGAS DE TRILHA		FAN OUT	TEMPO DE PROPAGAÇÃO	
		R	C		TRANSIÇÃO POSITIVA	TRANSIÇÃO NEGATIVA
SOMADOR	SOMA	6,4kΩ	76fF	3	28 NS	26 NS
	TRANSPORTE	-	-	1	13 NS	13 NS
COMPARADOR		-	-	1	11 NS	11 NS
"BUFFER" DE RELÓGIO	Ø	-	3,5pF	70	16 NS	23 NS
	$\bar{Ø}$	-	3,5pF	70	23 NS	16 NS
FLIP FLOP D DINÂMICO		6KΩ	700fF	-	13 NS	13 NS
INTERFACE TTL	ENTRADA	4KΩ	190fF	15	26 NS	26 NS
	SAÍDA	-	28pF	1 TTL-LS 1 CMOS	19 NS	17 NS

Figura 2.2

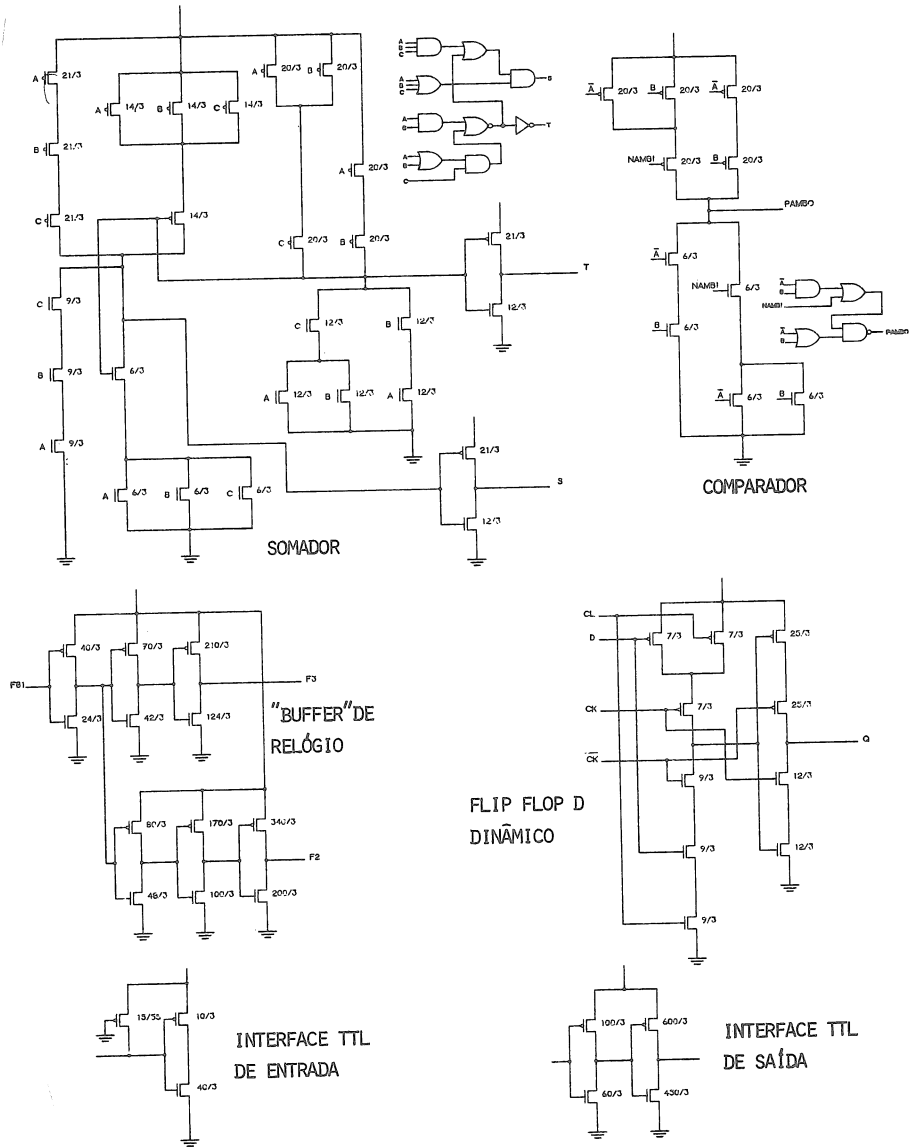

SOMADOR
COMPARADOR
"BUFFER" DE RELÓGIO
FLIP FLOP D DINÂMICO
INTERFACE TTL DE ENTRADA
INTERFACE TTL DE SAÍDA

Figura 2.1

CONCEPÇAO DE UM GERADOR DE SINCRONISMO - PROGRAMAVEL PARA VIDEOS TIPO VARREDURA

R.P.JACOBI*

SUMARIO

A concepção de um gerador de sincronismo programável para vídeos tipo varredura fixa é apresentada desde a definição das células básicas até o desenho das máscaras. O circuito é composto basicamente por dois blocos principais encarrecados da geração dos sinais de sincronismo vertical e horizontal, e por um conjunto de registradores auxiliares. Os blocos principais são compostos por contadores, por registradores programáveis e por comparadores. É apresentado o "lay-out" destas células básicas na tecnologia NMOS.

ABSTRACT

The design of a programmable syncronism Generator for raster scan CRTs is presented from the definition of the basic cells to the layout of the marks. This Circuit is basi cally made up of two syncronism generating bloks: one for the vertical and another for the horizontal signs, and a set of auxiliary registers. The main bloks are composed by NMOS implemented counters, programmable registers and comparators. The layout of these basic cells are shown.

* Engenheiro Eletrônico (UFRGS), aluno do Curso de Pós-Graduação em Ciência da Computação, UFRGS, Caixa Postal 1501, 90.000, Porto Alegre, RS.

1. Introdução

A utilização de terminais de vídeo como interface de comunicação entre homem e máquina tornou-se muito difundida em função da natureza essencialmente visual de aquisição de informações pelo homem. Diversos tipos de terminais de vídeo foram desenvolvidos com diferentes tecnologias. Como decorrência do barateamento dos componentes do hardware, como as memórias p. ex., os terminais de vídeo tipo varredura fixa atingiram uma relação desempenho/custo superior aos demais, na maior parte das aplicações. Outro fator que tem contribuido para a diminuição do seu custo é o surgimento de circuitos integrados dedicados ao controle da exibição da imagem, que agrupam diversas funções em uma única partilha de silício. Por esses motivos esse tipo de terminal de vídeo tornou-se o mais utilizado comercialmente.

O funcionamento do terminal tipo varredura fixa requer basicamente dois tipos de informação: o sinal de vídeo, que descreve a imagem,e os sinais de sincronismo, que possicionam a imagem corretamente na tela.O circuito apresentado neste artigo, foi proposto como um trabalho para a disciplina de Microeletrônica I, consistindo em um gerador de sinais de sincronismo para vídeos tipo varredura fixa, sendo o formato de exibição programável. A concepção foi feita em tecnologia NMOS com porta de silicio e pré-contatos, a ser fabricada em uma partilha tipo multiprojeto.

2. Função dos Sinais de Sincronismo

A imagem em um video tipo varredura fixa é formada pela incidência de um feixe de elétrons sobre a superfície interna da tela. Essa superfície contém um fósforo que emite luz ao ser excitado pelos elétrons. O movimento do feixe sobre a tela é similar ao movimento de escrita sobre uma página, da esquerda para a direita e de cima para baixo, sendo que um ciclo completo do feixe sobre a tela é chamado de campo. Assim como um texto em uma página,a imagem na tela do video também é circundada por uma margem . Durante este período, que inclui o retraço horizontal e vertical (retorno do feixe no fim de uma linha ou campo), o feixe é desligado.

A informação de quando o feixe deve retornar para iniciar uma nova linha ou um novo campo e de quando o feixe está margem ou na área de exibição da imagem deve ser fornecida ao terminal de vídeo. O circuito que fornece estas informações é o gerador de sincronismo. O sistema de exibição pode ser entrelaçado ou não. No sistema entrelaçado o feixe percorre as linhas pares, completando um campo, e depois as linhas ímpares, conforme mostra a figura 1. Desta forma é obtida uma definição maior com a mesma

frequência de vertical.

Em função dessas características, o circuito implementado gera os seguintes sinais:

a) PS H (pulso de sincronismo horizontal), que faz o feixe retornar para iniciar uma nova linha.

b) PSV (pulso de sincronismo vertical), que faz o feixe retornar para iniciar um novo campo.

c) HE (habilitação de exibição), que desliga o feixe nos períodos de "blank" (margem).

d) PSVE (pulso de sincronismo vertical entrelaçado), que, junto com o PSV, permite gerar o sistema entrelaçado.

e) CAMPO, que informa qual o campo em exibição, das linhas pares ou ímpares, no sistema entrelaçado.

3. Formatos de Exibição

O gerador de sincronismo concebido permite a utilização de diversos formatos de exibição. Estes formatos são estabelecidos em função do número de linhas exibidas na tela e do número de caracteres por linha. A unidade basica de tempo para o circuito é o período de exibição de um caracter na tela. Assim, a duração de uma linha e consequentemente de um campo, é um múltiplo desse período básico.

Os dados que definem o formato a ser utilizado e que podem ser programados são os seguintes:

a) número de caracteres contidos em uma linha

b) número de caracteres exibidos em uma linha

c) duração do PSH

d) atraso do PSVE em relação ao PSV.

e) número de linhas contidas em um campo.

f) número de linhas exibidas em um campo.

g) duração do PSV.

4. Arquitetura Interna e Funcionamento do Circuito

A arquitetura interna do circuito é mostrada na figura2. É utilizado um contador para registrar o número de caracteres contidos em uma linha. Este contador é ligado aos registradores que contém os dados rela-

tivos ao formato da linha. A cada registrador está ligado um comparador, que compara o valor do contador com o registrador. Esses registradores,com paradores e o contador tem 8 bits.

O contador vertical registra o número de linhas contidas em um campo. Está ligado a comparadores e registradores, que contêm as informações relativas ao formato do campo. Todos esses tem 10 bits, permitindo um formato de até 1024 linhas por campo.

Os comparadores detectam a coincidência entre o valor de um contador e o valor de um registrador, gerando um sinal que será utilizado para produção dos sinais de sincronismo. Desta forma, a programação do PSH, por ex., é feita gravando-se o número do caracter da linha onde começa o PSH e o número do caracter onde ele termina. Os contadores são resetados sempre que atingem o valor programado como número total de linhas/caracteres por campo/linha.

Os sinais de sincronismo são gerados por FFs tipo SR, que são controlados pelos sinais provenientes dos comparadores. O sinal CAMPO é gerado por um FF tipo TOGGLE, que é invertido a cada mudança de campo. A lógica de geração do PSVE é mostrada na figura 3. Utiliza o sinal PSV condicionado à informação de CAMPO. O atraso do PSVE em relação ao PSV é obtido a partir do sinal gerado pelo registrador/comparador que contém o número do caracter onde deve começar o PSVE.

5. Programação do CI

A escrita de um dado no circuito requer a habilitação do circuito (CS="Ø") e a ativação do sinal de escrita (WR="Ø"). O sinal SR é utilizado para selecionar o registrador de endereços ou os registradores programáveis. Um diagrama de tempos destes sinais é mostrado na figura 4. A gravação de um dado em um registrador envolve dois ciclos de escrita: No primeiro seleciona-se o registrador de endereços (SR="Ø") e carrega-se o endereço de registrador a ser gravado, e no segundo ciclo escreve-se o dado no registrador especificado (SR="1").

Como os registradores do bloco de geração de sinais de sincronismo para o vertical tem 10 bits e o barramento tem 8 bits, os 2 bits que faltam são armazenados no registrador de endereços durante o primeiro ciclo de escrita. Correspondem aos bits menos significativos do dado e são armazenados nos bits mais significativos do registrador de endereços. No segundo ciclo de escrita esses bits são gravados no registrador junto com os 8 bits do barramento.

O número total de registradores é 9, 5 para o horizontal e 4 pa

ra o vertical, e são endereçados conforme mostra a tabela 1.

6. Filosofia do Projeto

Procurou-se simplificar ao máximo as conexões e fazer o circuito de uma forma modular, baseado na repetição de poucas células básicas. Com isso minimizou-se o tempo gasto na concepção.

A simplificação nas ligações foi obtida pelo compartilhamento de um mesmo barramento pelo contador e pelos registradores e comparadores de um mesmo bloco, conforme figura 5.

Os barramentos de dados e alimentação foram dispostos ortogonalmente em relação aos sinais de controle, sendo a alimentação estruturada na forma de dois pentes imbricados, VCC e GND, abrangendo todo o circuito.

Foram utilizados PADs de entrada, saída, VCC, GND e relógio com gerador de duas fases não superpostas, fornecidos pelo fabricante. A planta baixa do circuito a nível de blocos, com a localização dos PAD's é mostrada na figura 6.

A regularidade dos blocos principais foi obtida pela repetição das células básicas do contador, registrador/comparador e registrador de endereços. O lay-out do circuito completo é mostrado na figura 7.

O fator de regularidade do circuito (FR), que é a relação entre o número total de transistores do circuito dividido pelo número de transistores realmente projetado, foi o seguinte:

$$FR = \frac{1675}{157} = 10,6$$

7. Células Básicas

A célula do contador é mostrada na figura 8.É composta por FF tipo mestre-escravo, com lógica de incremento na realimentação. A geração do vai-um é feita de forma complementada de uma célula para a outra. Nas células pares faz-se um NAND entre dado e o vem-um e nas células ímpares faz-se um NOR entre o dado negado e o vem-um negado, gerando-se um vai-um não negado para a célula seguinte. A lógica de incremento é feita através de um XOR entre dado e vem-um. Os FF são implementados por inversores realimentados por um transistor de passagem, e controlados por um relógio de duas fases não superpostas.

A célula registrador/comparador (figura 9) utiliza um FF estático, com um transistor tipo depleção na realimentação e um XOR que detecta a coincidência entre o valor do registrador e o do barramento. Quando

todos os bits de um registrador correspondem aos do barramento uma linha de controle em polisilicio é levada a "1", indicando a igualdade.

O registrador de endereços (figura 10) é igual a célula do registrador/comparador serem o comparador.

Os FF SR e TOGGLE utilizados pela lógica de geração dos sinais de sincronismo são mostrados na figura 11.

8. Conclusão

O circuito proposto implementa as funções básicas de geração dos sinais de sincronismo para um vídeo tipo varredura fixa. A flexibilidade decorrente da possibilidade de programação do circuito para diferentes formatos de exibição foi obtida de uma forma simples pelo compartilhamento do barramento dentro dos blocos principais. Essa característica permite ainda a expansão da capacidade do circuito pela inclusão de registradores nos blocos, implementando outras funções.

Procurou-se utilizar uma metodologia, descendente de concepção, analizando-se a estrutura do circuito a partir de seus blocos funcionais e descendo-se então até o nível das máscaras.

Tornou-se evidente durante a concepção a necessidade de ferramentas de software que auxiliem e automatizem o processo de concepção para garantir a confiabilidade do circuito concebido.

9. Bibliografia

/MEAD 80/ MEAD, Carver & CONWAY, Lynn. Introduction to VLSI systems. Addison-Wesley, 1980.

/MOT 82/ MOTOROLA, MICROPROCESSOR DATA MANUAL. MC 6845 CRT Controler Pag 4-457, 479. 1982.

/REIS 83/ REIS, Ricardo Augusto da Luz. Evaluateur Topologique predictif pour la generarion automatique des plano de masse de circuits VLSI. Grenoble, Institut Nacional Polytechique 1983.

/THOM 82/ THOMSON-EFCIS INTEGRATED CIRCUITS. Videotext CRT display processor. 1982.

/SUZ 81/ SUZIM, Altamiro Amadeu. Estude des partes a elements modilaires pour processeur monolothiques. Grenoble, Institut National Polytechnique, 1981.

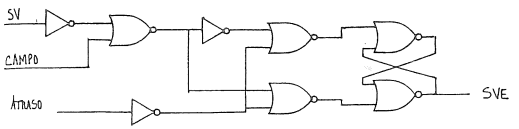


Figura 3: Lógica de geração do PSVE.

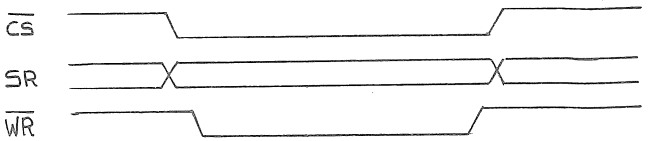


Figura 4: Diagrama de tempos para operação de escrita.

SR	REND						REGISTRADOR
	5	4	3	2	1	0	
0	X	X	X	X	X	X	Registrador de Endereço
1	0	1	1	1	1	1	Número Total de Linhas
1	0	1	1	1	0	1	Início Blank Vertical
1	0	1	1	0	1	1	Início PSV
1	0	1	0	1	1	1	Fim PSV
1	1	1	1	1	1	0	Número Total de Caracteres
1	1	1	1	1	0	1	Início Blank Horizontal
1	1	1	1	0	1	1	Início PSH
1	1	1	0	1	1	1	Fim PSH
1	1	0	1	1	1	1	Deslocamento do PSVE

Tabela 1: Programação do circuito.

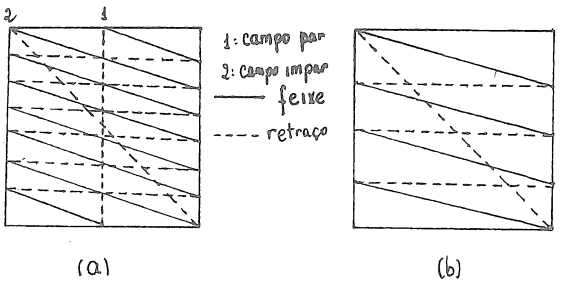


Figura 1: Sistema de exibição (a) entrelaçado e (b) não entrelaçado.

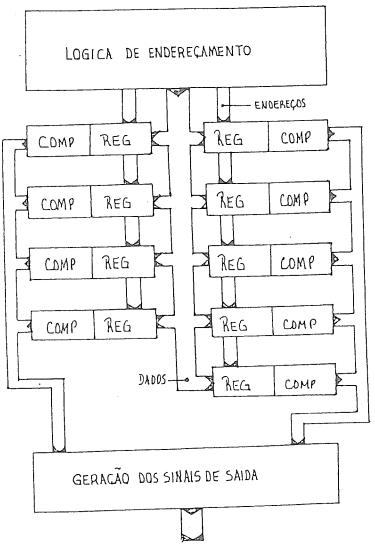


Figura 2: Arquitetura Interna.

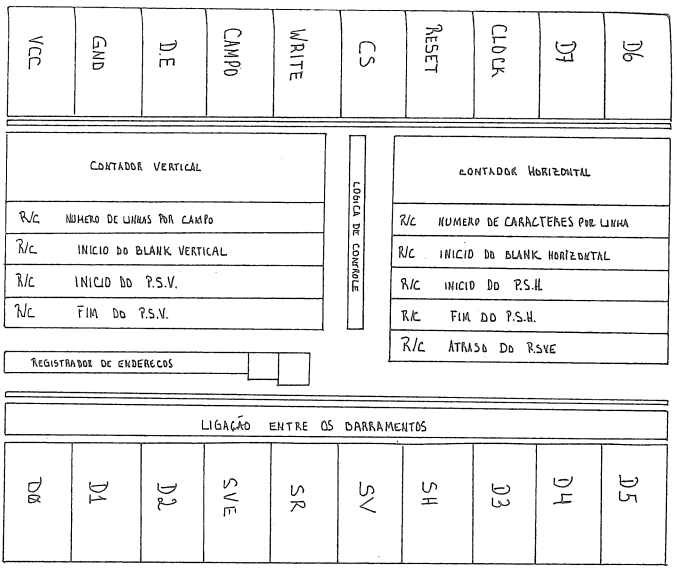


Figura 6: Planta baixa do circuito.

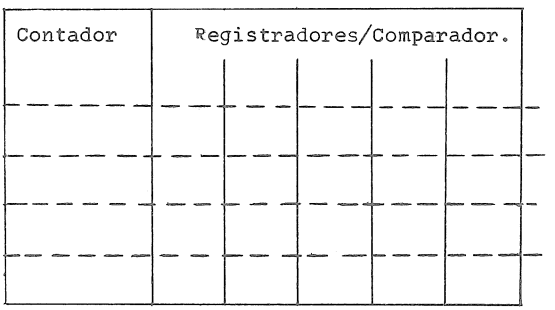


Figura 5: Barramento compartilhado.

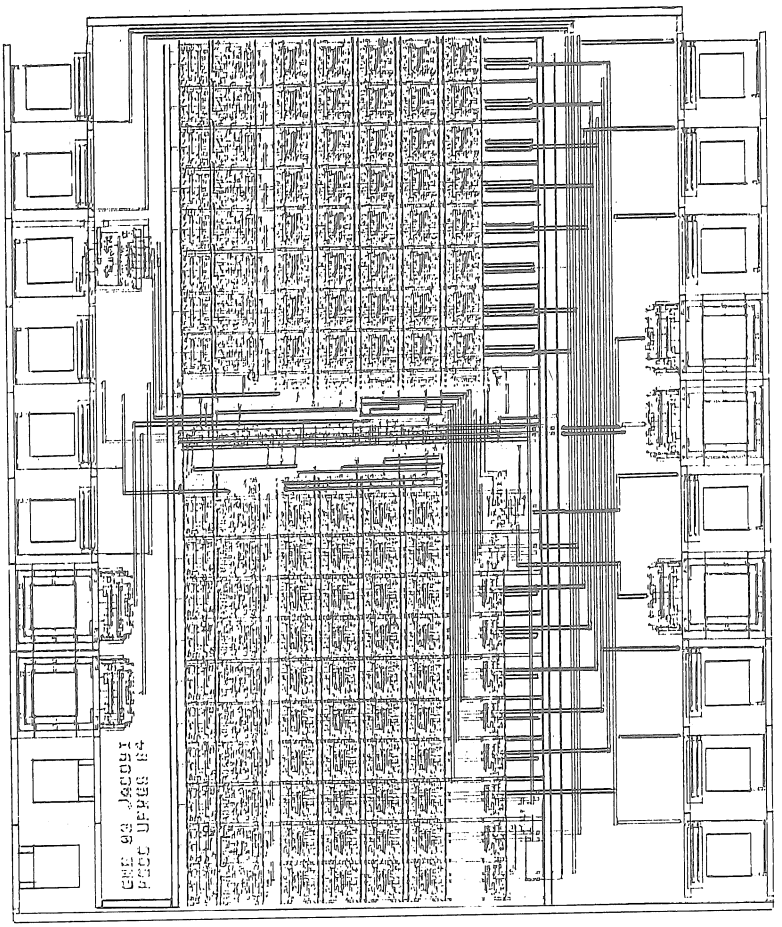


Figura 7: Layout final do circuito.

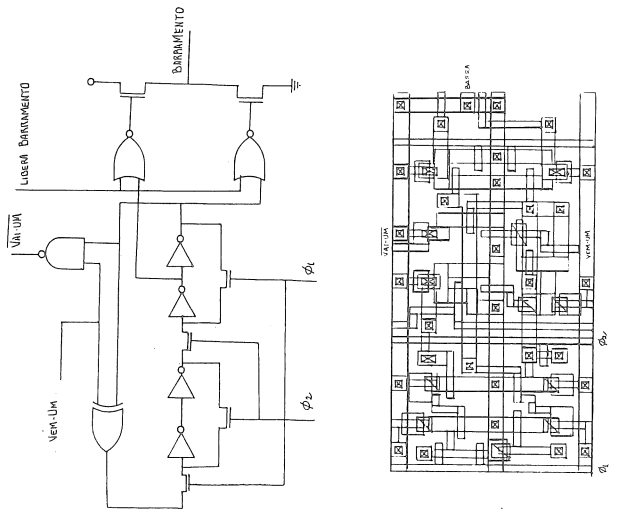


Figura 8: Contador: esquema logico e layout.

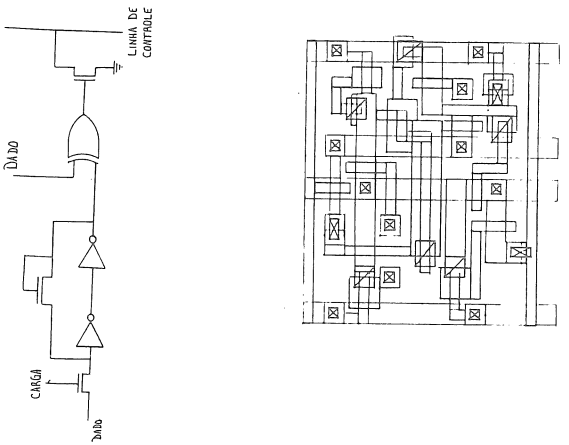


Figura 9: Registrador/comparador: esquema lógico e layout.

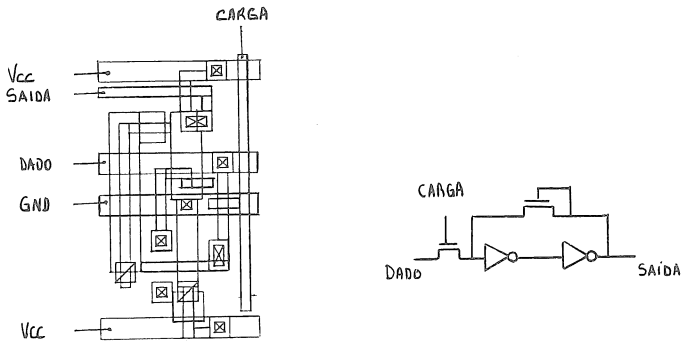


Figura 10: Registrador de endereços: layout e esquema lógico.

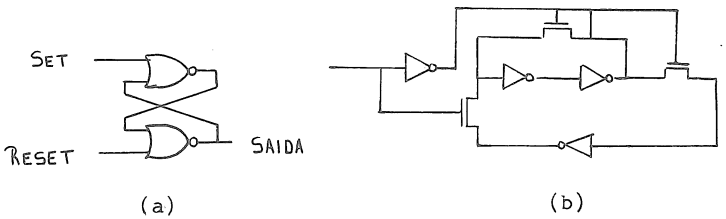


Figura 11: Flip-flops: (a) SR e (b) TOGGLE.

J.V.VALE NETO*

SUMÁRIO

Neste artigo será apresentado o projeto de um circuito "Detetor de Transição". Este é um circuito que gera um pulso para cada variação em sua entrada (0→1 ou 1→0).
Será mostrado como calculamos as dimensões de seus transistores. Através da análise de seus pontos críticos (pontos que podem comprometer o funcionamento do circuito) mostraremos que o circuito funcionará mesmo com variação de parâmetros (mobilidade e tensão de limiar).
Será mostrado, também, a simulação dinâmica do circuito.
Este circuito é parte do projeto de uma ROM-2K bits CMOS, que foi fabricada no LME-USP, no qual foi verificado o seu funcionamento.

ABSTRACT

In this paper is presented the design of a "Transition Detector". This circuit will generete a pulse when a transition occurn in this imput (0→1 or 1→0).
It will be shown how the transistor dimensions were calculated. We will show that the circuit will operate propely even when occuring parameter variation (mobility and threshold voltage) through analysis of the circuit critical points (which can cause circuit malfunction).
We will show the circuit dinamic simulation.
This circuit is a part of the project of a ROM-2Kbits CMOS, that was built in LME-USP, where its performance has been verified.

* . Engenheiro Eletricista, Escola Politécnica da USP, 1978.
. Bacharel em Física, Instituto de Física da USP, 1982.
. Mestre em Engenharia Elétrica, Escola Politécnica da USP, 1983.

Laboratório de Microeletrônica da Escola Politécnica da USP
Caixa Postal 8174 - 01000 - SÃO PAULO - SP
fone: (011) 815.93.22 ramal 310

INTRODUÇÃO

Neste trabalho será apresentado o projeto e as simulações de um circuito "Detetor de Transição" em CMOS.

Este circuito é uma das partes de uma ROM-2Kbits {1} (256x8) que foi fabricada no LME-USP.

Cada uma das oito entradas da ROM contará com um deste circuito.

A função deste circuito é gerar um pulso para cada variação de entradas da ROM (0-1 ou 1-0), este pulso será usado pela unidade de controle para iniciar um ciclo pré-carga/leitura que atualizará as saídas da ROM.

PROJETO

Na figura 1 mostramos o esquema simplificado do detetor de transição , este esquema será usado apenas para vermos o funcionamento do circuito.

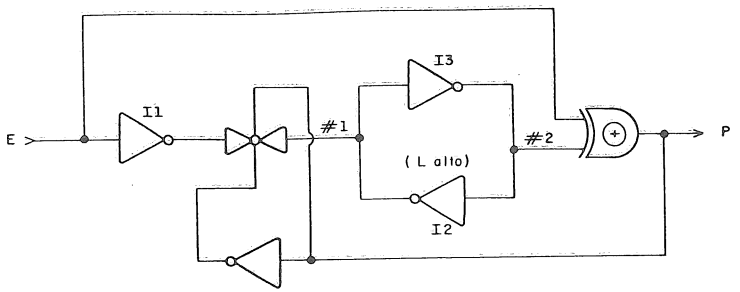


Figura 1 - Esquema simplificado do Detetor de Transição

Seu funcionamento vem a ser o seguinte:

. em condições estáveis a saída P estará em zero e o "gate de transmissão" estará não conduzindo, sendo que a tensão no nó 2 será igual a da entrada (E) ;

. ocorrendo uma transição na entrada (E), a tensão no nó 2 será(evidentemente) diferente da tensão de entrada (E) o que fará com que a saída P vá a "um" ;

. P indo a "um" fará com que o "gate de transmissão" conduza. Tendo o inversor I2 sido projetado "fraco" em relação ao sistema inversor I1 e "gate de transmissão", este imporá sua tensão no nó 1, fazendo isto, a ten

são no nó 2 voltará a ser igual a da entrada (E) e será sustentada pelo biestável composto pelos inversores I2 e I3; com isto a saída P voltará ao nível "zero" encerrando assim o pulso e o "gate de transmissão" deixará de conduzir, e, com isto o sistema voltará a condição estável até uma próxima transição.

É importante notar que: havendo uma transição na entrada (E) o circuito só voltará a condição estável após acontecer um pulso na saída P, isto torna esta implementação bastante segura em relação a taxa de variação da entrada (E) com o tempo.

O esquema completo deste circuito (como foi implementado) é mostrado na figura 2 , uma fotografia na figura 3 e a dimensão dos transistores na tabela 1. Seu funcionamento é o mesmo exposto acima, apenas teremos algumas pequenas sofisticações (em relação ao esquema simplificado) tais como: o diodo de proteção da entrada (T0) e o inversor na entrada (para minimizar a margem de ruído).

Transistores	TIPO	L(m)	W(m)	AD(m**2)	AS(m**2)	PD(m)	PS(m)
T0, T1, T4, T8, T10, T14, T16 , T18, T21, T25 , T28	N	6U	50U	256P	1P	64U	124U
T3, T22, T26	N	6U	200U	1248P	1P	188U	1U
T12	N	16U	88U	256P	1P	188U	1U
T2	P	6U	100U	560P	200P	108U	200U
T5	P	6U	132U	1008P	1P	132U	184U
T6	P	6U	254U	1728P	1P	248U	1U
T7, T13, T15, T17, T20, T24	P	6U	50U	256P	1P	64U	124U
T9, T27	P	6U	116U	816P	1P	116U	1U
T11	P	16U	88U	256P	1P	64U	1U
T19, T23	P	6U	200U	1248P	1P	188U	1U

Tabela 1 - Dimensões dos transistores do detetor de transição

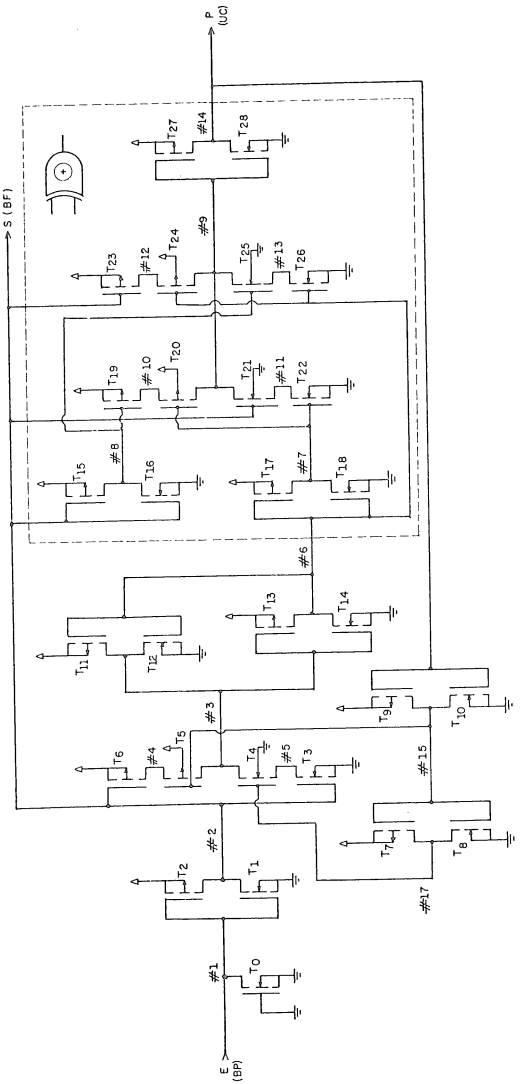


Figura 2 - Esquema do Detetor de Transição

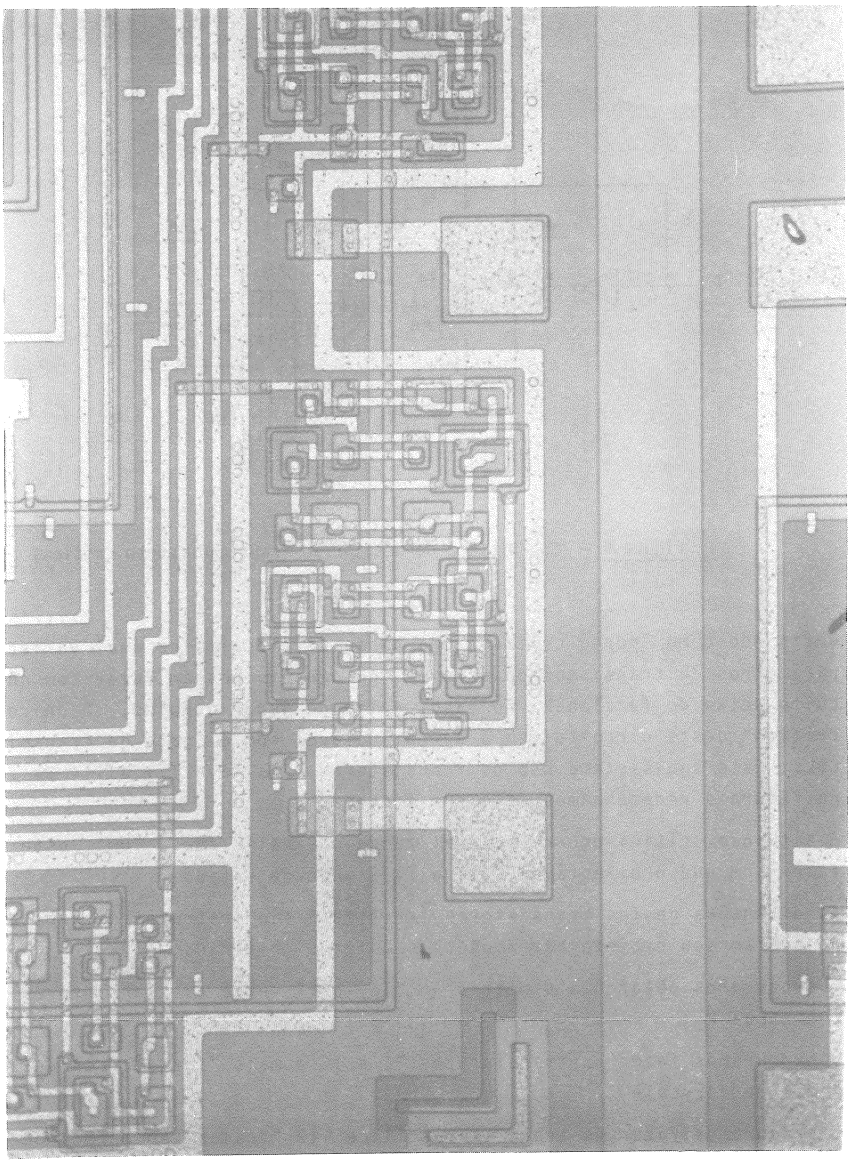


Figura 3 - Fotografia do Detetor de Transição

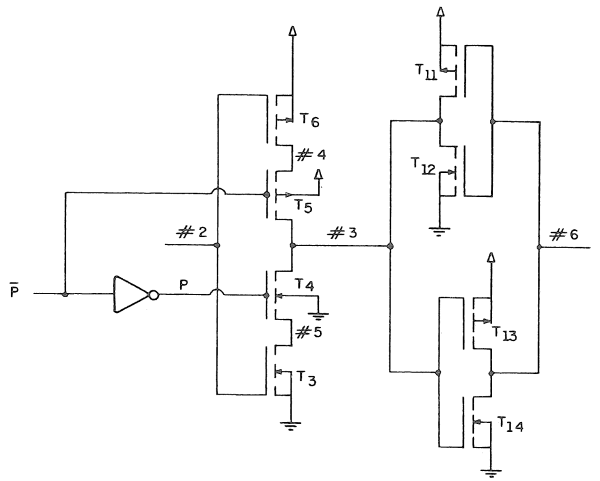


Figura 4 - Sub-Sistema do Detetor de Transição (ponto crítico)

Definindo como "ponto crítico" uma parte do circuito na qual variações nos parâmetros do transistor ou erro em seu projeto possam fazer com que o circuito deixe de funcionar, no caso, deixe de gerar o pulso. O único "ponto crítico" deste circuito vem a ser bi-estável com o "gate de transmissão" CCL. Este sub-sistema são os transistores T3 a T6 e T11 a T14 da figura 2, na figura 4 redezenhamos este sub-sistema.

A situação crítica ocorre quando: tendo o sinal P ido a "um", este deverá fazer com que o bi-estável (T11 a T14) mude de estado.

As dimensões destes transistores "críticos" são mostrados na Tabela 1, sendo que em seu projeto adotamos os seguintes critérios:

. procuramos obter uma simetria no "lay-out" entre os transistores canal N e canal P ;

. procuramos fazer os transistores T3 a T6 o mais "forte" possíveis em termos de corrente ;

. procuramos fazer os transistores T11 e T12 "fracos" em termos de corrente (L alto).

São duas as situações críticas, que poderão ocorrer:

a) Transição da entrada E de "um" a "zero".

Neste caso, quando o sinal P for de "zero" a "um" (fazendo conduzir os transistores T4 e T5) o circuito se encontrará no seguinte estado: os nós 2 e 3 estarão em "um" e o nó 6 em "zero". Logo, podemos ver que: os transistores T3 e T4 passarão a conduzir "concorrendo" com o transistor T11 , isto deve fazer com que a tensão no nó 3 caía. Devemos projetar o circuito de maneira a que a queda da tensão do nó 3 faça com que a tensão do nó 6 começe a subir (através do inversor formado por T13 e T14), esta tensão começando a subir fará com que o transistor T11 fique mais "fraco" (devido a queda em seu V_{gs}), isto fará com que a tensão no nó 3 caía mais e a tensão no nó 6 suba mais, sendo que esta realimentação se encarregará de acabar de inverter o bi-estável.

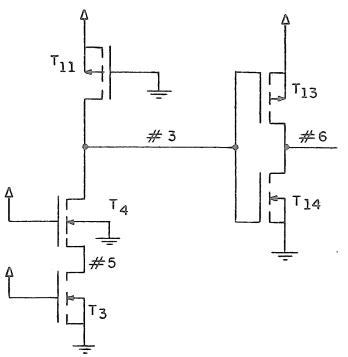


Figura 5 - Representação da situação crítica (a) do Detetor de Transição

Na figura 5 temos o esquema da situação exposta, sendo que cortamos a realimentação do nó 6 para a porta do transistor T11. Diremos que o circuito funcionará se a tensão do nó 6 for maior que 10% de V_{cc}.

Com os valores da Tabela 1 simulamos o circuito e obtivemos V3 = 0,36 V e V6 = 5,00 V , o que garante o funcionamento deste circuito.

Como este é um ponto crítico, também fizemos simulações permitindo variação de ± 20% nos parâmetros mobilidade e tensão de limiar dos transistores (com o objetivo de prever cargas no óxido e variação de temperatura). Tendo em vista estas variações, a situação de pior caso será a que tornar mais "forte" os transistores canal P e mais "fraco" os transistores canal N : logo

faremos a simulação com:

VTN = 1,54 V - variamos em + 20%
μn = 656 cm**2/V/s - variamos em - 20%
VTP = - 0,8V - variamos em - 20%
μp = 312 cm**2/V/s - variamos em + 20%

Com isto obtivemos V3 = 0,69V e V6 = 5,00V, o que indica que este circuito funcionará mesmo com estas variações de parâmetros.

b) Transição da entrada de "zero" a "um".

Neste caso, quando o sinal P for de "zero" a "um" o circuito se encontrará no seguinte estado: os nós 2 e 3 estarão em "zero" e o nó 6 em "um". Logo, os transistores T5 e T6 passarão a conduzir "concorrendo" com T12 , isto fará com que a tensão no nó 3 comece a subir. Devemos projetar o circuito de maneira a que a "subida" na tensão do nó 3 seja suficiente para fazer com que a tensão no nó 6 comece a cair. A queda desta tensão fará com que o transistor T12 fique mais "fraco" (devido a queda em sem V_{gs}) , isto fará com que a tensão no nó 3 suba mais e a tensão no nó 6 desça mais, até que o bi-estável mude de estado.

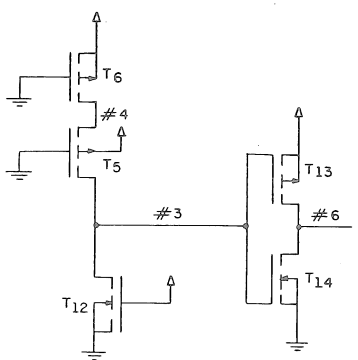


Figura 6 - Representação da situação crítica (b) do Detetor de Transição

A situação exposta é mostrada na figura 6 (podemos notar a semelhança com o caso anterior), também aqui cortaremos a realimentação do nó 6 para a porta do transistor T12, para fazermos os nossos cálculos. Neste caso, diremos que o circuito funcionará se a tensão no nó 6 for menor que 90% de Vcc, pois a realimentação se encarregará do resto da transição.

Fizemos as simulações deste sistema e obtivemos V3 = 4,01 V e V6 = 0,00 V, que satisfazem as nossas condições de funcionamento.

Aqui também fizemos simulações permitindo variações de ± 20% na tensão de limiar e na mobilidade dos transistores, sendo que a situação de pior caso e a que tornará mais forte os transistores canal N e mais fraco os transistores canal P, exposta a seguir:

VTN = 1,02 V - variamos em - 20%
μn = 984 cm**2/V/s - variamos em + 20%
VTP = -1,2V - variamos em + 20%
μp = 208 cm**2/V/s - variamos em - 20%

Com isto obtivemos V3 = 2,77 V e V6 = 0,11 V , o que indica que o circuito deve funcionar.

SIMULAÇÃO DINÂMICA

As cargas vista pelo circuito são mostradas na figura 7. Estas cargas foram obtidas levando em conta as capacitâncias da metalização de interconexão entre os blocos, as capacitâncias das portas que o canal alimentará e as capacitâncias e resistências das interconexões de silício policristalino (representação L1).

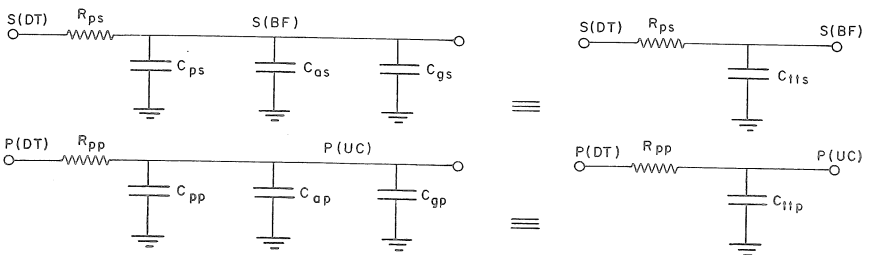


Figura 7 - Cargas vistas pelas saídas do Detetor de Transição

Na figura 7 teremos:

- R_{pp} e R_{sp} - resistência de tira de interconexão de silício policristalino ;
- C_{pp} e C_{ps} - capacitância da tira de interconexão de silício policristalino ;

. C_{ap} e C_{as} - capacitância da linha de alumínio ;

. C_{gp} e C_{gs} - capacitância das portas que o sinal alimentará.

Por exemplo o nosso bloco de entrada DET7 tem as seguintes cargas na saída S :

. tira de silício policristalino de 58 x 14 um**2 , como a resistência por quadrado do Si-Poli é de 50 oh (Tabela 3.1) temos R_{sp} = (58-2*7)*50/14 = = 157 oh, e C_{sp} = 58*14*3,38 e-4 = 0,27 pF (TOX e de 1 e-7m logo C_{ox} = = E_{ox}/TOX = 3,38 e-13/1 e-5 F/cm**2 = 3,38 e-4 pF/um**2 ;

. tira de alumínio de 375 * 10 um**2 , logo C_{ap} = 375*10*4,83 e-5 = 0,18 pF (pois COes = Eox/TOex = 4,83 e-5 pF/um**2) ;

. as áreas dos transistores de entrada de BF são A = (36*26-14*14+26*60-14*40)= = 1740 um**2 logo C_{gs} = 1740 * 3,38 e-4 = 0,59 pF.

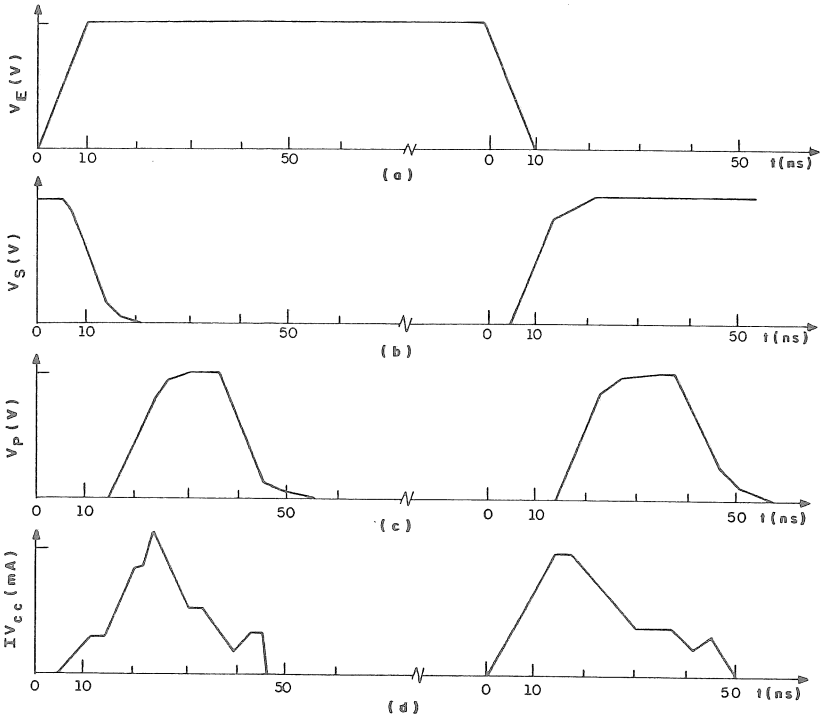
Com este procedimento calculamos todas as nossas outras cargas e, por isto, obtivemos a Tabela 2 (saída B) e a Tabela 3 (saída P).

BLOCO	TIRA SP (um**2)	R_{ps} (oh)	C_{ps} (pF)	TIRA AL (um**2)	C_{as} (pF)	PORTA (um**2)	C_{gs} (pF)	C_{tts} (pF)
DET0	58*14	157	0,27	3995*10	1,93	1740	0,59	2,79
DET1	58*14	157	0,27	3494*10	1,69	1740	0,59	2,55
DET2	58*14	157	0,27	2889*10	1,40	1740	0,59	2,26
DET3	58*14	157	0,27	2411*10	1,16	1740	0,59	2,02
DET4	58*14	157	0,27	1853*10	0,90	3220	1,09	2,26
DET5	58*14	157	0,27	1295*10	0,63	1740	0,59	1,49
DET6	58*14	157	0,27	837*10	0,40	1740	0,59	1,26
DET7	58*14	157	0,27	375*10	0,18	1740	0,59	1,04

Tabela 2 - Cargas da saída S do Detetor de Transição de acordo com a figura 3.13

BLOCO	TIRA SP (um**2)	R_{pp} (oh)	C_{pp} (pF)	TIRA AL (um**2)	C_{ap} (pF)	PORTA (um**2)	C_{gp} (pF)	C_{ttp} (pF)
DET0	58*14	157	0,27	2527*10	1,22	2012	0,68	2,17
DET1	74*14	214	0,35	2375*10	1,15	1480	0,50	2,00
DET2	90*14	271	0,43	1933*10	0,93	2012	0,68	2,04
DET3	106*14	329	0,50	1638*10	0,79	1480	0,50	1,79
DET4	122*14	386	0,58	1225*10	0,59	2012	0,68	1,85
DET5	138*14	443	0,65	667*10	0,32	1480	0,50	1,47
DET6	154*14	500	0,73	329*10	0,15	2012	0,68	1,57
DET7	170*14	557	0,80	791*10	0,38	1480	0,50	1,68

Tabela 3 - Cargas de saída P do Detetor de Transição de acordo com a figura 3.13



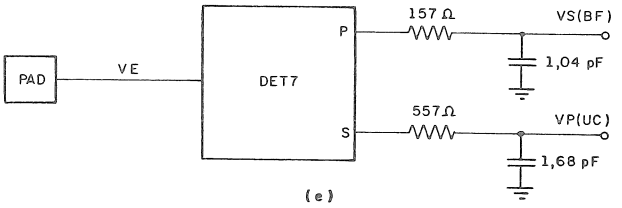


Figura 8 - Simulação do Detetor de Transição no domínio do tempo

a) Tensão de entrada VE (plano de contato)
b) Tensão de saída VS (vai para BF)
c) Pulso VP (vai para UC)
d) Corrente de fonte (IVcc)
e) Esquema usado

Simulamos este circuito no SPICE-II-G5 com as cargas do DET7 e com tensões de entrada (vindas do plano de contato) indo de 0V a 5V e de 5V a 0V em 10 ns (10 nano-segundo) e obtivemos as configurações de tempo mostradas na figura 8.

Por esta figura podemos ver que a largura de nosso pulso (definido nos pontos de 2,5V) será de 20 ns e 23 ns para transição da entrada de 0V a 5V e de 5V a 0V.

A carga consumida será de aproximadamente 40 pC por variação da entrada. Esta carga é a integral da corrente de fonte com o tempo e será usada para o cálculo da potência consumida.

BIBLIOGRAFIA

{1} VALE NETO, José Vieira - Tese de Doutoramento em andamento no LME-USP.

CONCEPCAO DE UM CIRCUITO DE COMUNICACAO VLSI PARA ARBITRAMENTO DE ARQUITETURAS MULTI-MICROPROCESSADORES: ABC M

DANTE AUGUSTO COUTO BARONE*

SUMARIO

Atualmente, as arquiteturas multi-microprocessadores vem sendo cada vez mais utilizadas, principamente em robótica e inteligências artificial. Estas arquiteturas necessitam que o barramento comum seja alocado adequadamente. Neste trabalho apresentamos a concepção de um circuito arbitrador VLSI o qual integra poderosas funções associadas à alocação do barramento.

ABSTRACT

Nowadays, the multi-microprocessor architectures are used in computer fields like robotics and artificial intelligence. These architectures must present an appropriate bus allocation function. In this work, we presente the design of a VLSI arbiter circuit. which integrates powerful functions associated to the bus allocation control.

*Engenheiro Eletrônico (UFRGS, 1978), Mestre em Engenharia Elétrica (USP, 1981), "Docteur Ingenieur" em Informática (INPG, 1984); concepção de circuito integrados, arquiteturas de computadores e multi-microprocessadores, Professore Adjunto da UFRGS, Caixa Postal 1501, Porto Alegre - RS - CEP 90000. Te.: 0512 - 218499.

INTRODUÇÃO

Os sistemas multi-microprocessadores tem se revelado uma solução atrativa desde aplicações em escritório até aplicações industriais.

Estes sistemas são organizados em torno de um barramento, de sorte que a informação trocada entre módulos mestre e seus recursos comuns, como memória e dispositivos de entrada e saída é suportada por um meio físico comum, chamado de barramento. Atualmente, a importância dos barramentos paralelos está aumentando consideravelmente. Alguns desses barramentos tornaram-se padronizados pelo órgão americano IEEE, (Institute of Electrial and Electronics Engineering). Esses barramentos por serem compartilhados necessitam uma função de controle para alocar o meio, evitando assim que módulos mestre pudessem ocupar simultâneamente o meio. Este controle é realizado freqüentemente por um circuito chamado arbitrador. Circuitos arbitradores assíncronos |1| |2| ou síncronos |3| foram desenvolvidos com o propósito de alocar conveniente o meio físico. Alguns desses arbitradores foram realizados através de circuitos integrados específicos.

Um importante inconveniente ligado à diversidade de barramentos paralelos decorre do fato de a técnica de arbitragem associada a cada barramento ser específica ao barramento em questão, não sendo compatível com outros.

Esta restrição impede a utilização de arbitradores mais poderosos em outros barramentos, o que seria altamente vantajoso, pois tais circuitos podem apresentar características tais como:

- permitir níveis cíclicos de prioridade
- mecanismos de controle de erro
- arbitragem distribuída

O circuito VLSI ABC M (Arbitre de Bus de Communication Multiprotocoles) desenvolvido no Centre National d'Études des Télécommunications de Grenoble, o qual é descrito neste artigo, foi concebido com os objetivos de propiciar uma função da arbitragem poderosa.

O circuito ABC M foi concebido para realizar a função de alocação em diferentes barramentos:

a) MULTIBUS (IEEE P796)
b) VME BUS (IEEE P1014)
c) SM BUS; barramento utilizado na arquitetura multi-microprocessadores heterogêneos SM90, desenvolvida no CNET - Paris e comercializada pela companhia francesa SEMS.

Entre os pesquisadores que podem se interessar por este projeto destacamos os engenheiros que estão desenvolvendo novas placas e novas arquiteturas para barramentos paralelos.

METODOLOGIA

Como nós consideramos para nosso estudo diferentes barramentos de comunicação paralelos e suas respectivas técnicas de arbitragem, as quais variam de acordo com o fabricante, nós escolhemos como hipótese de trabalho |4| de estudar a compatibilidade do circuito arbitrador mais poderoso com os outros dois tipos de barramento analisados.

O circuito arbritrador escolhido foi o circuito VLSI ABC 90 |5| o qual foi concebido para realizar a função de alocação da arquitetura SM 90 |6|. Este circuito apresenta algumas características, como capacidade de realização de arbitragem distribuída, incorporação de mecanismos de controle de erro e consideração de prioridades cíclicas no algoritmo interno de resolução de prioridade, as quais não se acham presentes nos circuitos arbitradores dos barramentos MULTIBUS, cuja arbitragem é realizada por componentes TTL (codificador e decodificador) |7| e VME (circuito MC 68452 |8|). Em realidade, estes últimos arbitradores são muito simples permitindo unicamente prioridades fixas associadas aos diferentes módulos mestre conectados ao barramento, além do que a arbitragem é realizada de forma centralizada.

Este estudo preliminar permitiu-nos de:

- definir as diferentes configurações arquiteturais (centralizada e distribuída) em que podemos colocar o circuito ABC 90, para que este possa realizar convenientemente a função de alocação em barramentos originalmente não compatíveis com este circuito: MULTIBUS e VME.

- definir a lógica discreta necessária a ser adicionada ao circuito ABC 90 para que o mesmo realize corretamente a função de alocação.

Nosso estudo de compatibilidade entre diferentes arbitradores e barramentos incluem uma análise aprofundada dos três barramentos analisados e de seus respectivos arbitradores.

Para realizar esta análise, nós consideramos três aspectos principais de compatibilidade:

- lógica, para demonstrar que os circuitos envolvidos podem dialogar corretamente.
- elétrica, para mostrar como os problemas de interface elétrica, como tipo de saída (coletor aberto, TTL) e capacidade de drenar corrente podem ser resolvidos |9| |10|.
- mecânica, para mostrar como os fios dos barramentos padronizados podem ser usados para realizar a função de alocação. A compatibilidade mecânica torna-se mais dificilmente obtida quando a arbitragem é realizada em um contexto distribuído, devido ao maior número de sinais envolvidos |9| |10|.

Esta análise aprofundada não será desenvolvida neste artigo; sendo encontrada em |4|.

Contudo, mostraremos para ilustrar esquematicamente a arquitetura de compatibilidade MULTIBUS - ABC 90 proposta (fig.1):

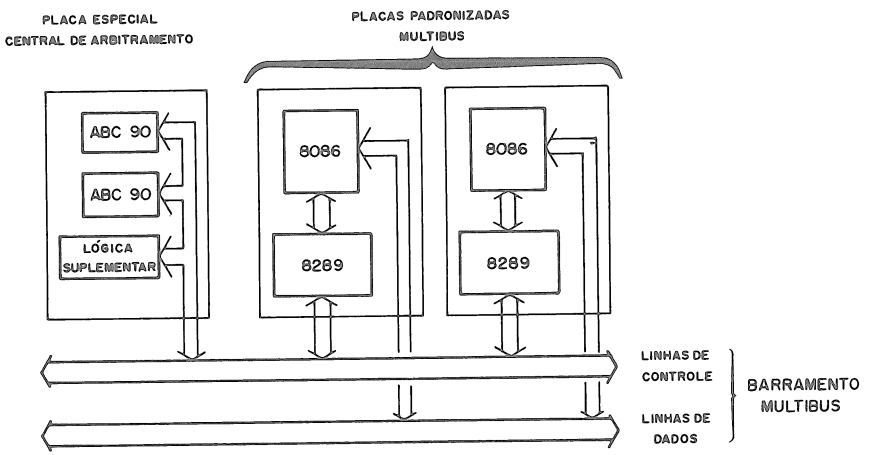


Fig. 1: Arquitetura discreta de arbitramento: MULTIBUS - ABC 90

A arquiterua mostrada consiste de algumas placas padronizadas MULTIBUS contento cada uma um arbitrador de baramento desenvolvido por INTEL (8289) e uma placa especial de arbitragem. Esta placa substitui a placa original contendo os circuitos responsáveis pela arbitragem (codificador (74LS138) e decodificador (74LS148). Esta placa especial, por nós proposta, contém alguns circuitos ABC 90 além de lógica suplementar necessária.
Neste artigo, chamamos de placa padronizada as placas manufaturadas e distribuídas comercialmente (como placas da família do microprocessador INTEL, 8086, da chamada família SBC) e de placa especial, uma placa desenvolvida especificamente pelo engenheiro de hardware de uma arquitetura multi-microprocessadores (como a placa de arbitragem centralizada contendo os circuitos ABC90, por nós proposta). Em nosso trabalho colocamos sempre os arbitradores de referencia (no caso o ABC 90) neste segundo tipo de placa, para não desperdiçarmos a compatibilidade com placas por ventura existentes no mercado.

Nesta arquitetura MULTIBUS - ABC 90, devido à simples substituição de uma placa especial por uma padronizada (contendo os circuitos codificador e decodificador), não encontramos nenhum problema de compatibilidade mecânica, enquando a compatibilidade elétrica é facilmente obtível |4|.

A placa central de arbitragem contém no mínimo igual número de circuitos ABC 90 que há placas padronizadas conectadas ao barramento MULTIBUS. Em realidade, há uma correspondência únivoca entre a posição física de cada placa padronizada no bastidor com um circuito ABC 90 localizado na placa especial central de arbitragem (parâmetro de posição física UC).

A necessidade do haver esta correspondência decorre de uma restrição do algoritmo interno do circuito ABC 90: os circuitos ABC 90 tendo sido concebidos para uma arquitetura onde a arbitragem é distribuída (arquitetura SM 90) somente o circuito arbitrador associado ao mestre que efetuou a última transferência de dados através do barramento deve "divulgar" qual o próximo mestre que deterá este direito, enquanto os circuitos codificador-decodificador "divulgam" sistematicamente.

Para tornar possível a compatibilidade lógica entre os circuitos ABC 90 e 8289, alguns componentes discretos foram introduzidos na placa especial.

VALIDAÇÃO

Devido ao fato de que nossas arquiteturas de compatibilidae cruzada são complexas (principalmente quando utilizamos circuitos ABC 90 para realizar uma arbitragem distribuída em arquiteturas de arbitragem originariamente centralizadas) devemos validá-las.

Decidimos em não implementar fisicamente as arquiteturas propostas devido à não disponibilidade de arquiteturas multi-microprocessadores compatíveis com os barramentos MULTIBUS e VME no local onde foi desenvolvido este projeto (CNET - Grenoble) e também devido aos tempos de desenvolvimento requeridos. Contudo várias possibilidades de validação podem ser consideradas.

Dentro de nossos propósitos, as ferramentas de Projeto Assistido pro Computador (PAC) apropriadas aos nossos objetivos de validação deveriam permitir:
a) possibilidade de descrição de módulos especificados por cronogramas (problema encontrado na descrição do comportamento do circuito 8289, do qual não conhecemos a implementação lógica interna).
b) possibilidade de simulação de uma arquitetura completa descrita como um conjunto de diferentes módulos.
c) possibilidade de realizar provas de coerência formal.

Tendo em mente estas considerações, tornou-se evidente para nós que ferramentas de validação teórica (condição c) são inadequadas para provar a corência de arquiteturas cujos

elementos são descritos por cronogramas. Conseqüentemente, escolhemos dois tipos de ferramentas de PAC:

- uma linguagem de alto nível do tipo HDL (FIDEL|12)|)foi utilizada para resolver a condição a descrita anteriormente.
- um simulador lógico (EPILOG |13)|) foi utilizado para resolver a condição b.

Devemos ressaltar que FIDEL e EPILOG são ferramentas de PAC compatíveis.

RESULTADOS DE SIMULAÇÃO

Os resultados principais obtidos através das simulaçãoes são descritos a seguir:

- o tempo de arbitramento é de cerca de 240ns para as arquiteturas centralizadas e cerca de 470ns para as arquiteturas de arbitragem distribuída (VME - ABC 90 e MULTIBUS - ABC 90)
- o relógio de resolusão de prioridade (sinal BCLK) do barramento MULTIBUS tem seu período aumentado para 400ns (o período mínimo para BCLK é de 100ns).
- a escolha do algoritmo de resolução de prioridade é livre para o usuário, podendo este configurar os módulos mestre concectados ao barramento em prioridade fixa ou rotativa. Além disso, pode-se ter também alguns mestre em prioridade fixa e outros em rotativa. Devemos lembrar que somente níveis fixos de prioridade são propostos pelas técnicas originais de arbitramento MULTIBUS ou VME.

No caso da arquitetura de arbitragem ser distribuída o algoritmo de cálculo das prioridades é mais complexo: se o último mestre a deter o barramento é configurado em uma placa padronizada, a divulgação do próximo mestre do barramento será feita pela placa especial central de arbitragem; se o último mestre a deter o barramento configurado em carta especial, tendo um circuito ABC 90 associado ao mesmo a divulgação será efetuada por este circuito. As arquiteturas distribuídas não serão discutidas em detalhes neste trabalho.

CONCEPÇÃO DE UM ARBITRADOR VLSI MULTIPROTOCOLOS

A segunda parte do projeto consistiu na concepção de um circuito integrado de comunição, chamado ABC M, projetado como uma extensão das especificações do circuito ABC 90 e baseado na análise dos resultados obtidos que viemos de mostrar.

O circuito ABC M deve realizar a função de alocação em diferentes arquiteturas multiprocessadores para três tipos de barramento: MULTIBUS, VME BUS e SM BUS. Por outro lado podemos configurar o ABC M como arbitro de barramento local, o qual serve para conectar recursos de E/S e de memória de um microprocessador (mestre único do barramento local) como controladores de disco, controladores de DMA e etc. Este último tipo de aplicação não é possível com o circuito ABC 90.

Como nós vimos anteriormente, os principais obstáculos, para a utilização diretamente compatível do circuito ABC 90 com os barramentos MULTIBUS e VME são:

- "timing" de divulgação do direito de acesso ao barramento (veredito) = somente um circuito ABC 90 publica o veredito a cada vez (aquele associado com o último mestre possuidor do barramento). Esta dificuldade obrigava a placa central de arbitragem comportar tantos arbitradores quanto o número de placas padronizadas concectadas ao barramento (ver figura 1).
- o número e o "timing" dos bits de publicação do veredito (quatro para o circuito ABC 90 e somente um para os barramentos MULTIBUS e VME).
- o fato que o circuito ABC 90 realiza a sua função de arbitragem somente se um pedido local de acesso ao barramento realizado pelo microprocessador associado ao arbitrador ocorre (sinal DAB). Desde que o sinal DAB é ativado, o circuito ABC 90 genera automaticamente um sinal, chamado BREQ, que é transmitido ao barramento, o qual ativa simultaneamente a função de arbitragem em todos os ABC 90 conectados ao barramento. Como este sinal não encontra análago nos fios padronizados pelos barramentos MULTIBUS e VME, o mesmo é gerado através de componentes discretos.

Para resolver estes problemas nós analisamos diferentes algoritmos possíveis para o circuito ABC M. A implementação da melhor solução algoritmica por nós encontrada se caracteriza por:

1) - aumentar o número de pinos para 40 (o circuito ABC 90 apresenta 28 pinos)
   - os doze pinos suplementares consistem de:
     a) dois pinos que servem para definir o modo de funciomento do circuito:
        - RWD ("release when done")= o mestre libera o barramento assim que termina a utilização do mesmo
        - ROR ("release on request")= o mestre só libera o barramento, se um pedido mais prioritário é formulado
        - modo SM 90 = o circuito ABC M funciona tal qual um arbitrador ABC 90 em uma arquitetura SM 90.
     b) oito pinos que apresentam o veredito codificado (um fio dos oito é ativado somente a cada vez)
     c) dois pinos para desacoplar as linhas de alimentação no interior do circuito (um pino suplementar para VDD e outro para VSS).

2) - permitir de haver ou não uma divulgação permanente do veredito. No modo de funcionamento SM 90, cada circuito ABC M tal como o circuito ABC 90 não publica sistematicamente o veredito, ao invés dos dois outros modos onde todos os circuitos ABC M presentes na arquitetura publicam o veredito em paralelo. Isto pode ser implementado através de um modificação lógica do circuito ABC 90.

3) - introduzir uma lógica suplementar para respeitar as diferentes características da utilização do arbitrador multiprotocolos ABC M em três tipos diferentes de barramento, no tocante à publicação do veredito tais como:
   - diferentes "timing"
   - diferentes número de bits associados

4) - introduzir um autômato suplementar de controle para resolver o problema da geração do sinal de pedido de arbitragem (BREQ) (o circuito ABC 90 apresenta quatro automatos de controle implementados por um formalismo próximo das redes de Petri).

SIMULAÇÃO DO CIRCUITO ABC M

Uma vez que as modificações aportadas ao circuito ABC 90 transformando-o em arbitrador multi-protocolos são complexas, como viemos de descrever, as mesmas devem ser validadas por computador.

Nós utilizamos a mesma metodologia de validação que foi utilizada para a validação das arquiteturas de arbitramento discretas construídas a partir do circuito ABC 90.
- descrição em FIDEL para o circuito 8289 e a unidade de pedido de barramento (bus request unity) no caso dos barramentos MULTIBUS e VME, respectivamente.
- descrição e simulação de toda arquitetura em EPILOG.

Os resultados obtidos podem ser resumidos a seguir:
- o tempo de arbitramento é constante e de cerca de 250 ns para cada uma das arquiteturas compatíveis propostas: centralizada ou distribuída. No caso do barramento VME este tempo é de cerca de 60ns. No caso do barramento MULTIBUS, o retardo associado ao circuito ABC M é traduzido por um aumento do período do relógio de resolução de prioridade (BCLK) que fica em 400ns.

No entanto, devemos ressaltar que este maior tempo de arbitramento não causa uma lentidão na transferência efetiva de dados através do barramento, pois o circuito ABC M tal como o arbitrador ABC 90 permitem a realização de arbitragem "cachê", ou seja, durante a ocupação do baramento por um módulo mestre, os arbitradores determinam qual o mestre que terá o direito de ocupar o meio, assim que acabar a transferência em curso.
- a configuração de prioridade (fixa ou rotativa) de cada módulo mestre apresenta total liberdade aos utilizadores do circuito ABC M.

Além dos resultados de simulação nós podemos concluir através da figura 2, que mostra uma arquitetura centralizada MULTIBUS, que além das vantagens acima enumeradas o circuito ABC M se constitui em solução econômica comparada com a utilização correspondente do circuito ABC 90, mostrada na figura 1. Esta característica material (número de

componentes discretos associados à função de arbitragem) é ainda mais acentuada no caso das arquiteturas distribuídas.

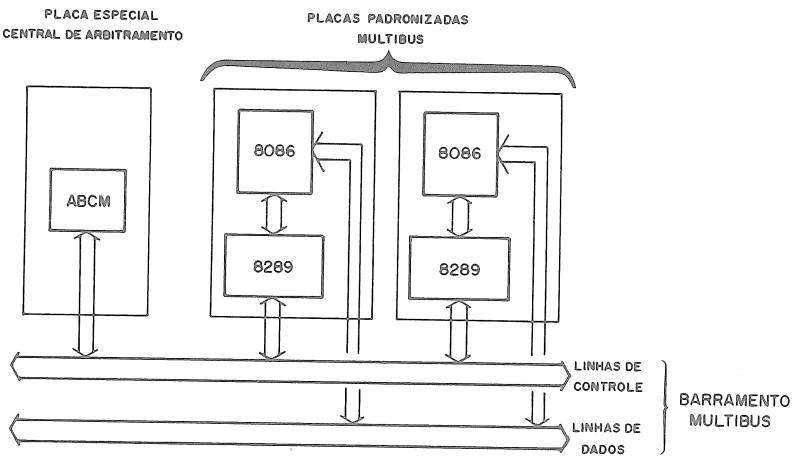


Fig. 2: Arquitetura integrada de arbitramento: MULTIBUS - ABC M

LAY-OUT DO CIRCUIT

Na implantação do ABC M em Silício, nós tivemos em mente um compromisso entre diferentes características:

- facilidade de corte do plano de massa original do circuito ABC 90
- problema de colocação de novos trilhas internas e "routing"
- introdução de novos blocos lógicos

O plano de massa obtido para o circuito ABC M, é cerca de 20% superior ao correspondente plano de massa do arbitrador ABC 90, contendo cerca de 10% a mais de transistores.

O circuito ABC M foi concebido em tecnologia NMOS, comprimento do canal de 3,5u m, nas regras de desenho da indústria francesa MATRA-HARRIS e do CNET - Grenoble com as seguintes ferramentas de PAC:

- banco de dados específico CASSIOPEE |14|:
  - algumas partes foram desenhadas utilizando uma linguagem simbólica do tipo stick M |15| e outras foram automaticamente sintetizadas (como autômato de controle suplementar e alguns PLA).
- sistema CALMA:
  - realizamos a conferência das regras de desenho (DRC: design rules checking) e a ge

ração das fitas contendo o conteúdo de cotas para posterior fabricação das máscaras.

CONCLUSÕES

Se compararmos a utilização do ABC M com as técnicas de arbitragem de origem para os barramentos MULTIBUS e VME, concluímos que o circuito ABC M permite:

- consideração de níveis cíclicos de prioridade
- mecanismos de controle de erro e recuperação em caso de erro
- redundância de serviços
- arbitramento distribuído

Devemos enfatizar que apenas algumas das vantagens descritas acima podem ser obtidas com a solução ABC 90.

Além de adicionar funções poderosas, a solução totalmente integrada (realização de arbitramento através do circuito ABC M) é mais econômica e fiável.

Além do fato de ser o circuito ABC M uma alternativa atrativa para barramento paralelos existentes, esperamos que nosso estudo possa ser utilizado como referência a futuros estudos de compatibilidade entre diferentes barramentos e diferentes arbitradores.

Finalmente, cumpre ressaltar que atualmente um minucioso estudo de marketing vem sendo efetivado na França, onde foi concebido o circuito ABC M, com vistas à fabricação industrial do mesmo em grande escala.

BIBLIOGRAFIA

|1 | PLUMMER, W.W. Asynchronons arbiters. IEEE Trans. on Computers, C-21 (1): 37-42, Jan. 1972.

|2 | PEARCE, R.C.; FIELD, J.A.; LITTLE, W.D. Asynchronous arbiter module. IEEE Trans. on Computers, C-24 (9): 931-2 Sept. 1975.

|3 | SCORDALAKES, N.E. Arbiter handles shared resources for multiprocessor. Computer Design, 20(12): 175-6, Dec. 1981.

|4 | BARONE, D.A.C. Conception d'um circuit integré arbitre de bus de communication multiprotocoles: ABC M. Grenoble, Institut National Polytechnique de Grenoble, 1984. (Tese de doutorado)

|5 | OLIVE, C. & ROUQUIER, D. Conception d'um circuit integré arbitre de bus ABC 90. In: CONGRESS de 1' AFCET, Lille, oct. 1982.

|6 | FINGER, U & MEDIGUE, G. Architecture multiprocesseur; l'exemple de la SM 90. Minis et Micros, oct. 1982.

|7 | INTEL CORPORATION. Multibus data book. 1983

|8 | MOTOROLA SEMICONDUCTORS. MC68452 advance information. 1983

|9 | INTEL CORPORATION. INTEL multibus specifications. 1981.

|10| MOTOROLA SEMICONDUCTORS. VME bus specification manual M68KVMEB (D1) microsystems. s.d.

|11| INTEL CORPORATION. 8289 bus arbiter technical notes. July 1979.

|12| HAZEM, M.; MAZARE, G.; POIZE, M.; PUISSOCHET, A. Functional modelling for logic simulation. In: INTERNATIONAL CONFERENCE IN COMPUTER DESIGN, New York, oct. 1984. Proceedings. New York, IEEE, 1984.

|13| EPILOG manual de reference. Jan. 1979.

|14| BEYLS, A. M. et alii. A design methodology based upon symbolic lay-out and integrated CAD tools. In: DESIGN AUTOMATION CONFERENCE, 19., Las Vegas, June 14-16, 1982. Proceedings. New York, ACM, 1982. p. 872-8.

|15| MEAD, C. & CONWAY, L. Introduction to VLSI systems. Reading, Addison-Wesley , 1980.

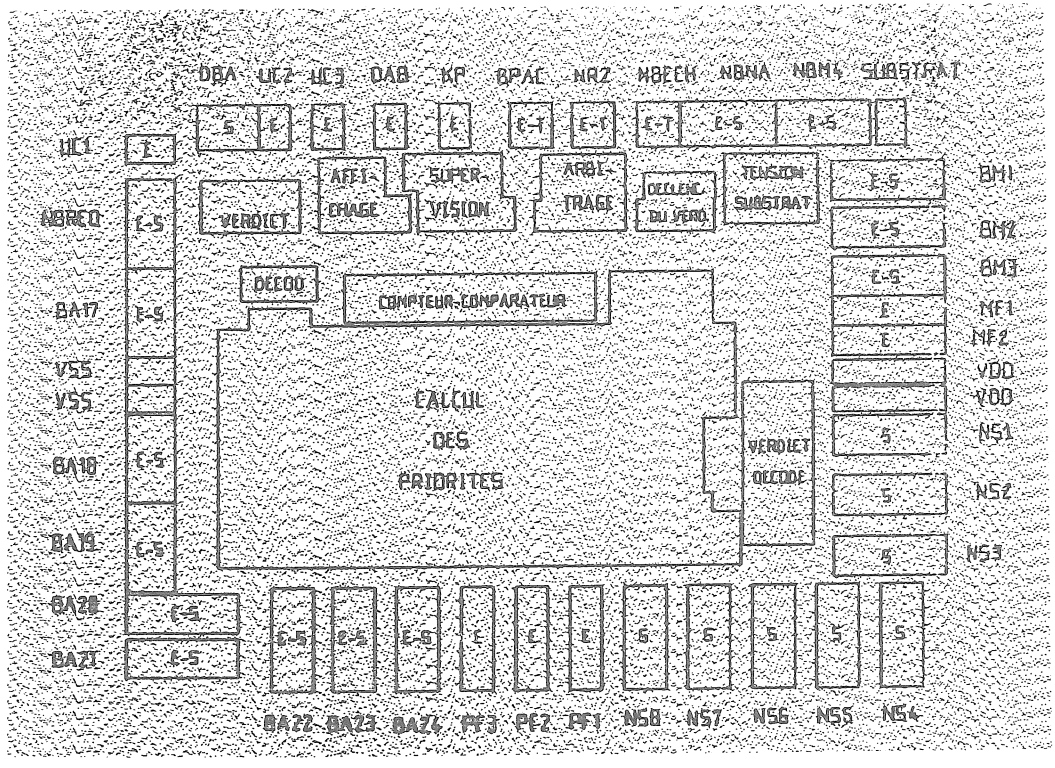


Figure 40 - plan de masse de l'ABC M

J.M. LARAIA*
A.L.P. JANZON**
L.O.F. GONÇALVES***

SUMARIO

Este trabalho descreve um circuito digital de deteção multifrequencial, próprio para o reconhecimento de tons em Sistemas PCM. O dispositivo emprega um algoritmo inédito, baseado em transformada discreta de Fourier, e é implementado com tecnologia NMOS de 4 um, com porta de silício.

ABSTRACT

A digital multifrequency detector chip suitable for tone recognition in PCM systems is described. It uses a novel algorithm based on discrete Fourier transform and is implemented in 4 um Si-gate NMOS.

* Engenheiro eletrônico (ITA-1979); especificação e projeto de circuitos integrados; Vértice Sistemas Integrados, R. Carlos Guimarães 248/102, Campinas - SP, CEP 13100, tel. 51-2366.

** Engenheiro elétrico (UNICAMP - 1978); especificação e projeto de circuitos integrados; Elebra Microeletrônica, R. Bogaert 326, V. Vermelha, São Paulo - SP.

*** (Ver outro trabalho de autoria de Luiz Otávio Fontenelle Gonçalves)

**** Trabalho apresentado na Custom Integrated Circuits Conference (CICC-85) realizada em Maio de 1985 em Portland, OR - EUA.

INTRODUCTION

A well-known problem in many communication systems is how to execute the detection of tones, or in other words, how to decide whether or not during a certain time interval a specific frequency has been received.

It was not until the mid-seventies that the availabity of inexpensive memories enable filtering to be performed by mathematical calculation, offering an alternative to the passive and active filters used as the only resource to the date. With the pressures of futher reducing costs, new algorithms were proposed in order to save hardware, employing techniques such as correlation, zero-crossing count and fast Fourier transform.

In this decade, with the advancements of integrated circuit technology and greater densities of components made possible on the silicon, there is a definite trend towards the integration of whole systems in a single chip. In the analog field this resulted among other products in complete banks of filters integrated using switched-capacitor techniques. But the most remarkable advances occured with the digital circuits. One of the great advantages of the digital approach is that the same hardware may be designed to perform effectively a large variety of functions. Thus, the need of a wide range of application and real-time processing capabilities has led to the creation of commercial "single-chip" digital signal processors (DSP's) as a family og components which combine general purpose architecture and number-crunching hardware. However, there is a penalty in power consumption, cost and package size that could be avoided in most applications by a component tailored to execute more specific functions. With an architecture oriented to perform an otimized algorithm, it is frequently possible to get a much more efficient component, or even to make feasible functions not covered by state-of-the-art comercial DSP's.

The custom DSP implementation described in this paper, although primarily designed for a multifrequency cod (MFC) signaling receiver used in a digital exchange, is sufficiently versatile to be employed in a number of applications envolving detection of frequencies in PCM systems.

SPECIFICATIONS

The R2 signaling specifications that are important to the design are sumarized below : the dual tone signal consists of the linear addition of two voice frequency signals. Both are selected from either a group of frequencies called the "High Group", or from another group called the "Low Group", depending whether the exchange in question is the originator or the receiver the communication. Each group has six frequencies, as shown in Table 1.

	High Group	Low Group
1	1380 Hz	1140 Hz
2	1500 Hz	1020 Hz
3	1620 Hz	900 Hz
4	1740 Hz	780 Hz
5	1860 Hz	660 Hz
6	1980 Hz	540 Hz

Tabela 1 - R2 signaling frequencies

The detector should recognize signals with deviation up to ± 10 Hz and reject any signal with deviation greater than ± 60 Hz. The required dynamic range is 30 dB. If the amplitude of each signal do not differ by more than 5 dB for adjacent frequencies, or 7 dB for non-adjacent frequencies, detection of both signals is madatory. If one amplitude is more than 20 dB below the other one, then the correspondent signal should not be recognized.

Signals lasting less than 7 ms should be rejected and time spent to detect a signal plus the time needed ot detect its end must not total more than 80 ms. The allowed signal to noise ratio is 20 dB.

DESIGN APPROACH

The input signal used for the algorithm development is a sequence of n instantaneous samples of a cosinusoidal wave with radian frequency ω, unitary amplitude and initial phase θ. This signal is represented by a function of time defined below, parameterised in ω, θ and n:

$$g_{\omega,\theta,n}(t) = \sum_{i=1}^{n} \delta(t-iT)\cos(\omega t+\theta) \qquad (1)$$

where $\delta(t)$ is the Dirac delta function. The Fourier transform of $g_{\omega,\theta,n}(t)$ in the frequency f_D is given by:

$$\hat{g}_{\omega,\theta,n}(f_D) = \int_{-\infty}^{\infty} g_{\omega,\theta,n}(t)\, e^{-j\omega_D t}\, dt =$$

$$= \sum_{i=1}^{n} |\cos(\omega iT+\theta)|\, e^{-j\omega_D iT} \qquad (2)$$

Based on some properties of equation (2), it was theoretically developed an adaptive algorithm which presents some very desirable characteristics for tone detection, including:

- wide dynamic range
- low false detection rate
- operation with poor signal-to-noise ratio
- capability of detecting superposed signals
- insensitivity to signal phase
- good immunity to harmonic and intermodulation distortion
- low sensitivity to rounding errors
- easy digital implementation

An extensive work of simulation confirmed these characteristics and was useful to optimize the algorithm, choosing the most adequate constants in order to obtain better performance and minimize the hardware. Simulation results for a situation where there are two tones and white gaussian noise present in the input signal, can be seen in Figure 1.

Precision is relaxed wherever possible in order to save hardware, but simulations have shown that the overall effect in performance is negligible because of the intrinsic algorithm properties. One of the approximations in done with the amplitude of a quadrature pair. The digital implementation of the relation (3) is done very easili with the crude approximation (4):

$$Z = \sqrt{x^2 + y^2} \qquad (3)$$

$$Z' = |x| + |x| \qquad (4)$$

The incoming PCM data has to be internally linearized. If it were maintained the full PCM resolution, the linearized samples should have 12 bits, including the signal bit. However there is a rounding and the 3 least significant bits are eliminated, saving three sets of arithmetic cells and data buses. Dynamic range is still kept in excess of 40 dB.

For the circuit operation a sequence of sine and cosine samples of each detection frequence must be supplied by a external ROM. These samples are hardlimited to a three-level representation, as shown in Figure 2.

That can be expressed in harmonic series by the equation (5):

$$f(\omega t) = \frac{2\sqrt{3}}{\pi} \left(\cos \omega t - \frac{\cos 5\omega t}{5} - \frac{\cos 7\omega t}{7} + \frac{\cos 11\omega t}{11} + \ldots\right) \qquad (5)$$

It should be noted that the third harmonic is not present, and the fifth harmonic amplitude is 14 dB below the fundamental component. This is important to guarantee that signals containing frequencies that are submultiple of the detection frequency do not wrongly activate the detector, because of the aliasing.

The trigonometric samples have to be multiplied by the converted PCM input. Its two-bit quantization makes possible to accomplish this operation with a simple 8 x 2 multiplier.

IMPLEMENTATION

Once the algorithm and a compatible technology have been chosen, the architecture has come as a space x time trade-off. The serial design approach allows smaller chip size, while the parallel techniques usually imply greater area on the chip, and consequent higher power dissipation, but yields higher speed and easier control.

The algorithm was tested in a breadboard and needed about 50 TTL IC's, among MSI and and SSI parts. In an attempt to convert the TTL design to an NMOS one, several alternatives had to be considered. The final design involved the use of a 16-bit parallel processor to meet the speed constraints for the real-time process. Even though the data buses are 16-bit wide, the chip does not occupy a large amount of area, because of two main reasons: the care taken during the algorithm definition and logic mininization, and the strategy of using metal control lines crossing perpendicularly short poly data buses, embebded in handcrafted cells with standard height matching the RAM I/O pitch. This last feature allows a surprisingly compact and efficient layout of the arithmetic block.

The final architecture exhibited a highly planar topography, and there are no interconnection lines that cross each other. This led to a remarkably regular and modular layout.

As shown in Figure 3, the chip has a serial-to-parallel input register, connected to an 128 x 8 bit ROM that performs the PCM A-law to linear conversion. μ-law conversion is possible with a change int the programming mask. The ROM interfaces with the arithmetic block, which receives also the trigonometric samples. The output is transmitted through an 8-bit serial register. The control block is synchronized by external clock signals and has the function of keeping the aritmetic block performing its functions in the correct sequence. It is composed of a 16-bit adder and combinational circuitry. Although the overall control can not be presented here in detail, a two-phase clock was utilized and the timing of the process is sketched in Figure 4, where it is shown the division of the PCM sampling interval in 8 time slots. One detection frequency is processed in each one of the last seven time slots, while the first one is reserved for common calculations. Internally the slots are subdivided in 16 cycles, each having 2 clock

periods. It is the cycle decoding that effectively comands the sequence of arithmetic operations.

In the arithmetic block (Figure 5) the data are represented in two's complement notation. The 16-bit data buses are shared by two registers, an adder, an 8 x 2-bit multiplier, a module extractor, a magnitude comparator and a 26-word RAM. Considering the cyclic chacarteristics of the process and the fixed clock rate, it was selected a dynamic RAM with three transistors per cell. An equal output was added to the comparator, improving the circuit visibility and making easier the design of the test pattern. The whole block occupies approximately half of the total chip area.

The chip layout was totally handcrafted whit the aid of a symbolic layout editor.

PERFORMANCE AND APPLICATIONS

For the circuit implementation it is used a 4 micron silicon gate NMOS technology, with depletion load transistors and single layers of metal and polysilicon.

Table 2 sumarizes the basic hardware characteristics of the digital detector. The micro photograph can be seen in Figure 6.

Originally the algorithm and circuit were conceived to meet the R2 interregister signaling system. On the other hand, as those specifications are relatively stringent, and the chip interface and internal structure were designed with the necessary flexibility , it is expected that the detector should work without problems under other signaling systems, such as R1, Nº 5, Socotel and DTMF (Touch-Tone ^{R}).

Package	18-pin DIP
Chip size	3.0 x 2.8mm
Number of transistors	5500
Power sypply	5 volt (single)
Power consumption	50 mW (typical)
I/O interface	TTL compatible
Technology	4 µm Si gate NMOS

Table 2 - Hardware Characteristics

Up to seven independent frequencies in the voiceband may be programmed simultaneously for detection, using an external ROM with a sequence of sine and cosine sample of each frequency. However, it is recommended for practical reasons that the resoluction is set at a convenient value, limiting the number of samples. The number of sampling points needed for each detection frequency is given by (6):

$$n = \frac{f_s}{R} \qquad (6)$$

where: n = number of sampling points

f_s = sampling frequency (Hz)

R = detector resolution (Hz)

If f_s is an exact multiple of R, the last sample will coincide with the first one and it will be supplied a continuous sampling of the detection frequency, what is desirable for good operation. An adequate resolution value is 20 Hz, that will demand 400 sampling points in a PCM system sampled at 8 kHz. Taking advantage of the simetry properties of the sinusoidal function with very little extra hardware, it is possible in this case to use 100 x 4 bits of external memory for each frequency.

In the Table 3 there is a sumary of the basic performance characteristics of a signalling receiver based on the described chip.

Sampling rate	8 kHz
Clock rate	2.048 MHz
Detection range	0 to 4000 Hz
Approximate bandwidth	70 Hz
Output refresh interval	25 ms

Table 3 - Performance characteristics

ACKNOWLEDGEMENTS

The authors acknowledge the contribution of Celso de Oliveira, who made the basic definition of the algorithm. The IC implementation was made using installations and CAD tools of American Microsystems Inc. by a Telebrās - AMI Joint Development Team.

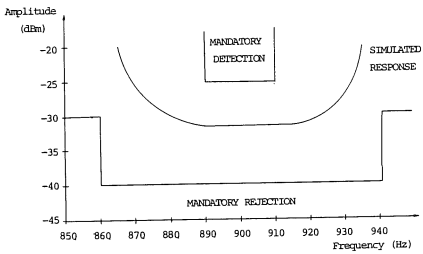


Fig. 1 Simulated response for 900 Hz detector with fixed 780 Hz tone at - 20 dBm and white gaussian noise at - 40 dBm.

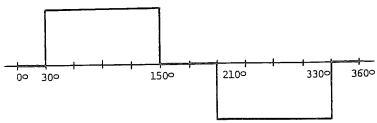


Fig. 2 Sine/cosine quantization.

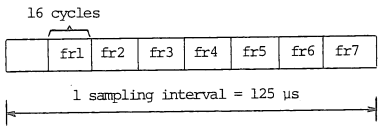


Fig. 4 Internal timing.

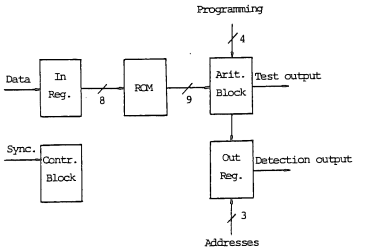


Fig. 3 Block diagram.

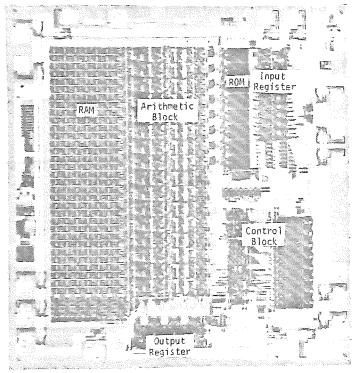
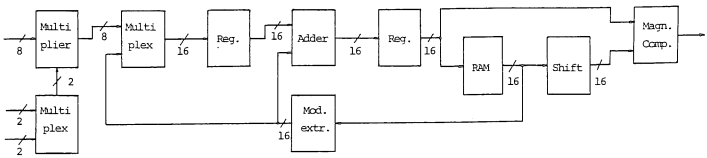


Fig. 6 Microphotograph of the digital detector



ALBERTO BASTOS DO CANTO FILHO *

SUMÁRIO

O objetivo deste trabalho é a apresentação das facilidades oferecidas pelo editor gráfico LACSUZ. O objetivo deste sistema é o auxílio ao projeto de microcircuitos através da utilização de um hardware de baixo custo (microcomputador pessoal). Seu uso será feito por estudantes de cursos de graduação, em disciplinas de introdução ao projeto de circuitos integrados.

ABSTRACT

The aim of this paper is to present the LACSUZ graphic editor features. The goal of this system is to support the integrated circuits design activity in a low coast hardware (personal microcomputer). It will be used by under-graduation students in introductory classes of I.C. design.

* Engenheiro Eletrônico (UFRGS-1981); Concluindo mestrado em computação na área de computação gráfica; Curso de Pós-Graduação em Ciência da Computação - UFRGS; Av. Oswaldo Aranha, 99 - Porto Alegre - RS CEP 90.000.

** Projeto parcialmente financiado pelo convênio UFRGS - FINEP - CPGCC número 52.84.0208.00

Sem dúvida alguma um "capítulo a parte" nos projetos de microcircuitos são as suas ferramentas de auxílio. Entre elas a computação gráfica ocupa uma posiçao de destaque.

Projetar é uma atividade essencialmente intelectual, mas sua materialização é normalmente feita através de desenhos. Neste contexto, um "editor gráfico" eficiente, que facilite fazer desenhos, corrigi-los, armazená-los e tirar cópias, aumenta enormemente o tempo disponível de um projetista de microcircuitos.

Preocupar-se com ferramentas de apoio para a microeletrônica é, acima de tudo, a preparação de um ambiente propício de pesquisa, onde os pesquisadores tenham o conforto de dedicar-se à sua especialização a maior parte de seu tempo.

O sistema aqui descrito parte do princípio que um sistema de CAD para microeletrônica não é necessariamente oneroso. Procurou-se desenvolvê-lo num microcomputador de 8 bits (Apple-2) com o intuito de dispor de ferramentas suficientemente acessíveis para o ensino de microeletrônica em cursos de graduação.

Para atingir este objetivo, foi feita uma divisão em 3 subsistemas:

a) subsistema de edição de arquivos;
b) subsistema de manipulação de arquivos;
c) subsistema de controle dos periféricos.

Com isto, procurou-se dar ao usuário condições de:

a) fazer objetos (desenhos) com o uso do computador;
b) armazenar os objetos em disco;
c) editar (alterar) objetos armazenados;
d) obter cópias em papel de objetos existentes;
e) fazer programas que acessem os arquivos para gerar novos formatos.

Sua definição foi feita sempre tendo em vista o futuro usuário: profissionais de microeletrônica. Isto levou à utilizaçao de retas e retângulos como elementos de trabalho, bem como a um método de interação voltado para esta aplicação.

A seguir serão descritas brevemente algumas facilidades deste sistema.

A principal preocupação no desenvolvimento deste sistema foi a interação com os usuários de microeletrônica, o que determinou quais seriam os recursos implantados.

2.1. Indicação de pontos

A primeira facilidade foi o elemento de trabalho: retângulos ou retas. Ambos definidos por dois pontos:

Segmentos de reta definidos por seus pontos extremos e retângulos definidos pelos pontos de uma das suas diagonais.

O modo como foi definida a interação com o usuário baseia-se em dois elementos, correspondentes aos dois pontos de trabalho:

a) ampulheta: posicionada no ponto ao qual o usuário está se referindo (o usuário sempre opera com a ampulheta);

b) borboleta: posicionada no segundo ponto de trabalho.

Neste sistema é possivel "editar" a ampulheta, isto é, definir a posição da ampulheta, e com isto especificar as coordenadas de um ponto. Após a edição deste, há uma troca: a borboleta é colocada no ponto recém editado e a ampulheta passa a apontar para o segundo ponto, permitindo assim a sua edição.

Com isto, é possível ir editando os dois pontos até que se esteja satisfeito com suas posições,que irão definir a reta ou retângulo.

As facilidades já implantadas dizem respeito à maneira pela qual o usuário pode "editar" a ampulheta. A interação com usuário pode ser feita de dois modos:

a) via "Paddle";
b) via teclado.

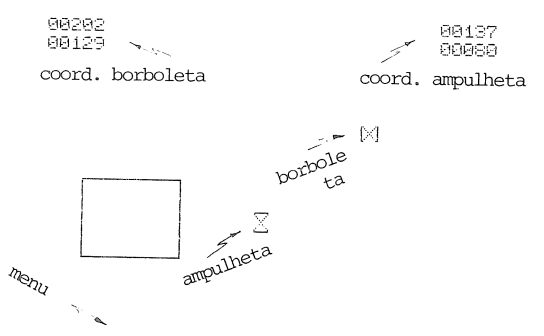


Figura 1: Indicação de dois Pontos

O "paddle" é um protótipo constituído por 3 botões e 2 potenciômetros (portanto de baixo custo). Através dos botões, o usuário se refere às opções do "menu" de operações. Com os potenciômetros é possível mover um cursor na tela para efetuar apontamentos.

Assim, o usuário pode fornecer as coordenadas da ampulheta de diversas formas:

a) apontamento: através dos potenciômetros, move-se o cursor para a posição desejada (até que os valores de coordenada, que aparecem no canto do monitor sejam satisfatórios).

A "sensibilidade" dos potenciômetros pode ser ajustada através dos botões de ajuste fino ou grosso;

b) adjacência: é possível, indicar pontos adjacentes a segmentos de retas ou retângulos já existentes. (os 2 pontos adjacentes a um segmento de reta são seus extremos; os 4 pontos adjacentes a um retângulo são seus vértices). Neste modo de operação, o usuário poderá apontar (com os potenciômetros) a linha à qual pertence o ponto adjacente desejado;

c) especificação via teclado: é possível fornecer via teclado os valores de X e Y, tanto absolutos, como relativos (somar valores à coordenadas atuais).

Os critérios de adjacência e especificação relativa de coordenadas foram inseridos tendo em vista as especificações de microeletrônica, onde deve-se guardar uma distância mínima entre os elementos de trabalho.

2.2. Facilidades de Visualização

Janela de seleção refere-se a uma parte do universo com a qual se deseja operar, pois nem sempre se deseja operar com todo o circuito. Através desta facilidade, pode-se especificar qual a parte do circuito com que se está trabalhando e somente esta será apresentada na tela. O resultado é uma vista de detalhe, portanto maior e de mais fácil interação.

Janela de exibição especifica que será usada somente parte da tela.

Através do conjunto seleção-exibição de janelas, informa-se "qual a parte do circuito" deverá aparecer em "qual parte da tela".

Eventualmente, pode ser importante operar simultaneamente com mais de um detalhe, ou com a vista de todo o circuito e dois detalhes, etc. Para satisfazer este tipo de exigência, foram implantados 4 conjuntos seleção-exibição de janelas. é possível expecificá-los, bem como torná-los

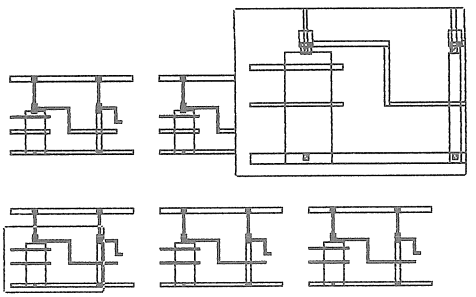


Figura 2: Duas janelas, mostrando todo e detalhe.

Com esta estrutura, fica facilitada a localização em circuitos muito grandes, bem como a especificação de linhas longas, como as de alimentação, por exemplo.

2.3. Facilidades de Armazenamento

Paralelamente ao editor gráfico, existe um sistema de arquivos, que permite armazenar os circuitos construídos.

Este é constituído por um diretório, sobre o qual é possível efetuar operações sobre arquivos.

Os arquivos são usados para armazenar em disco todas as informações gráficas necessárias: fazer um novo desenho é criar um novo arquivo; alterá-lo é alterar este mesmo arquivo. A importância destes arquivos reside, mais do que em sua capacidade de guardar uma imagem, na possibilidade de se gerar novos formatos para confecção de máscaras, por exemplo. Esta conversão de formato pode ser feita com programas relativamente simples. Outra possibilidade interessante é a construção de uma biblioteca de células: módulos que, interligados, podem constituir objetos mais complexos.

3. DESENHO ESTRUTURADO

A segunda parte do sistema editor de microcircuitos está em fase de acabamento, e diz respeito a "desenho estruturado".

Aqui os elementos de trabalho não são retas ou retângulos: são

circuitos. É possível, por exemplo, projetar uma determinada célula CELØ como um conjunto retas e retângulos e posteriormente criar a célula CEL1, constituída por três células CELØ dispostas lado a lado. A seguir, pode-se criar a célula CEL2, constituída por duas células CEL1 e uma célula CELØ girada de 90º, por exemplo. E assim sucessivamente.

A filosofia hierárquica adotada evita o dispêndio de esforço repetitivo. Neste contexto, é permitido, no sistema de arquivos, listar os pais de uma célula (todos os arquivos que fazem referência direta a uma célula), os ancestrais (pais, pais dos pais, etc...) e os filhos (aqueles aos quais uma célula faz referência), com a finalidade de localização dentro da estrutura hierárquica.

4. CÓPIA EM PAPEL

O periférico para "hard copy" é uma impressora gráfica EI 6010 (Mônica) da elebra. Com ela, pode-se obter em papel, cópias dos desenhos desejados.

5. CONCLUSÃO

Embora ainda não esteja concluido,o sistema editor de microcircuitos se apresenta como uma grande possibilidade para o ensino de microeletrônica nas universidades brasileiras. Num país onde a escassez de recursos para o ensino e pesquisa é um grande fator de limitação tecnológica, deve-se procurar soluções dentro deste contexto, que permitam contornar as dificuldades financeiras através de facilidades de mais baixo custo.

J.A.S. BORGES*

SUMÁRIO

O projeto de circuitos integrados está intimamente ligado ao uso de um editor gráfico onde se possa fazer operações de criação e montagem das partes componentes. Este artigo mostra algumas características de dois editores gráficos para projeto VLSI que foram produzidos no NCE-UFRJ.

O primeiro editor é o EDMOS, que já vem sendo utilizado desde 1983 e utiliza um microcomputador nacional (SDE-42) com vídeo gráfico e disquete como estação do trabalho.

O segundo editor é o EDCI, um editor hierárquico que visa acompanhar o projeto desde a alocação inicial de áreas até a montagem completa.

ABSTRACT

The project of IC's always needs a graphic editor, where one can do operations of creating and assembling the lay-outs. This paper shows the two NCE-UFRJ's circuit editors.

The first of them is EDMOS, that is in use since 1983. This editor uses a brazilian microcomputer (EBC-SDE-42) with a videographic interface and diskette, very cheap, which is an important factor at nationwide context.

The second is EDCI, an hierarchical editor which supports the project since area allocation until complete assembly.

* Informático (1980 - UFRJ); Processamento Gráfico; Estações de CAD; Universidade Federal do Rio de Janeiro, Núcleo de Computação Eletrônica, Caixa Postal 2324, Rio de Janeiro, CEP 20001.

As fábricas de circuitos integrados recebem a especificação de um chip na forma de uma linguagem de descrição da geometria, na sua maioria a CIF (Caltech Intermediate Format) Assim, a maneira mais rudimentar de especificar um circuito integrado para a fábrica é desenhá-lo em papel milimetrado e depois traduzir manualmente cada elemento da figura para esta linguagem. O desenho traduzido é entrado através de um editor de textos e depois copiado para uma fita magnética, que é enviada ao fabricante.

Infelizmente isto é completamente inviável: a quantidade de elementos que compõem um circuito pequeno é da ordem de dezenas de milhares, o que implica num trabalho sobre-humano. Basta que um erro seja cometido no processo de codificação para que o chip não funcione (e que sejam perdidos dezenas de milhares de dólares, que é o que a fundição de silício cobra por um protótipo).

Mesmo no processo de alteração, o método manual apresenta muitos inconvenientes. Uma simples operação de ajuste, ou seja, puxar um trecho do desenho um pouco mais para a direita, envolve a alteração da posição de todos os elementos do desenho compreendidos naquele intervalo.

Uma das abordagens mais aceitas no projeto de Circuitos Integrados hoje, é o uso das chamadas "células padrão" (Standard cells, biblioteca de células) que já foram projetadas e não apresentam erro. A união destas células é trabalhosa pois envolve o problema da diagramação do espaço físico disponível no chip e da codificação das ligações entre os elementos (sempre muito numerosas) mesmo que muitas células se encaixem perfeitamente (por abutment).

Todo este processo, altamente sujeito a erros, cria a necessidade de uso de papel para verificação da montagem do layout, geralmente em plotter ou impressora gráfica. Porém até mesmo essa visualização é difícil, pois o desenho é sempre muito intrincado.

SOBRE TERMINAIS E ESTAÇÃO DE CAD

Embora seja certamente possível editar layouts (especialmente de células) em terminais alfanuméricos (Lewis-81) a prática demonstra que o uso de terminais gráficos na edição de circuitos integrados é muito melhor, principalmente pelo aspecto visual e pelo tamanho do circuito que é possivel se ver simultaneamente numa tela.

Entre as características desejáveis da tela do terminal, deveríamos a princípio especificar um terminal que tivesse uma tela de 1024 x 1024 pontos, cursor retangular, 256 cores simultâneas, suportado por um computador de médio porte.

Esses requisitos são difíceis de serem conseguidos no Brasil, especialmente porque a produção de terminais gráficos é ainda muito incipiente. É porém fácil de se conseguir um microcomputador com tela gráfica colorida de resolução 240 x 400 (no nosso caso, foi um EBC-SDE42 com interface videográfica), que no caso, fica transformado em uma estação local para edição de circuitos.

O uso desta estação local é muito interessante, pois além do baixo custo, tem a vantagem da disponibilidade (em centros de computação universitários a disputa pelo uso do computador é enorme). Um micro é suficientemente barato para que possamos ter várias unidades, ao invés de um terminal muito caro, possibilitando assim que vários projetistas trabalhem ao mesmo tempo (o que é fundamental em turmas de cursos sobre projetos de VLSI.) Por outro lado, o armazenamento local de parte do circuito que está sendo editado e a possibilidade de processamentos locais (como verificação das regras de projeto) da célula que está sendo editada, faz com que só se precise mesmo usar o computador grande para os processos que envolvam maior necessidade de poder computacional (por exemplo, simulações do circuito).

A esta estação gráfica deve estar conectado um plotter de mesa e/ou uma impressora gráfica preto-e-branca, para hardcopy, pois a cópia em papel é necessária para que um projetista possa ir estudar em sua sala, liberando a estação para outra pessoa.

A estação local deve estar ligada ao computador principal através de uma linha de comunicações comum (9600 bauds), mas a frequência de transferência de informações é relativamente pequena (da ordem de 10 a 100 kbytes para arquivos CIF).

Assim, resumindo, na estação fazemos a edição das células e seu DRC. Usamos a estação como terminal gráfico simples para fazer a planta baixa, montagem do circuito e ligações entre as células, e usamos o computador principal para a verificação final e simulações, além da plotagem completa em plotter de alta velocidade.

VISÃO GERAL DO EDMOS

Mostraremos agora o EDMOS, que é o editor gráfico de células. O projeto deste editor foi influenciado por dois editores muito conhecidos: CIFSYM (Lewis-81) e CAESAR (Ousterhout-80). Em termos de processamento é semelhante ao primeiro; em termos de visualização, semelhante ao segundo.

O EDMOS é um editor que usa a memória local do video gráfico para armazenamento do layout (96kbytes). Não existem estruturas de dados intermediárias, ou seja, o layout é feito sobre a própria memória da tela, acendendo ou apagando os pontos convenientes como se estivéssemos colorindo um papel. Quando o desenho está completo, a memória de tela é varrida, extraindo-se o desenho.

Obs.: Foi feita posteriormente uma segunda versão do EDMOS que não mais utiliza estas características intrínsecas do equipamento utilizado. Esta nova versão incorpora para armazenamento das informações uma estrutura de dados armazenada através de um processo semelhante a memória virtual e que pode ser transportada facilmente para qualquer equipamento.

O tempo de projeto do EDMOS foi extremamente curto (um mês, um projetista), principalmente pela filosofia de uso direto da memória da tela, e pelo fato do processo de edição, em si, ser um processo puramente geométrico, com pouca necessidade de estruturas de dados sofisticadas.

Entre as principais facilidades do EDMOS, temos a criação e retirada de retângulos coloridos (box de CIF), ajuste de trechos, cópia, zoom, replicação de trechos, espelhamento, rotação, etc., que provêm a maioria das facilidades necessárias para a edição de células.

Entretanto o EDMOS não é capaz de editar layouts maiores que o tamanho da tela (240 x 400 lambda), embora com alguns macetes de projetista, isto tambem seja possível. Entretanto, poucas células são projetadas com mais do que este tamanho (não se contando com células que são compostas de outras células). Tanto as facilidades de montagem do layout quanto a hierarquia de células são providas pelo outro editor, EDCI, descrito mais adiante.

ALGUNS COMANDOS DO EDMOS

a) o cursor e sua movimentacao

O cursor é um retângulo branco, que pode ser movido, expandido e encolhido. Com este cursor delimitamos a área que nos interessa. O cursor é alterado usando exclusivamente o teclado. Este processo é mostrado em (Borges-83).

b) Modos de operação

Criamos dois modos de operação: rápido e lento. O primeiro é usado basicamente para mover o cursor e introduzir novos retângulos no desenho, teclando-se sempre uma tecla de cada vez. O modo lento é iniciado teclando-se RETURN e a seguir um mnemônico da função a executar. Neste modo, os comandos são interativos, ou seja, perguntam explicitamente todos os parâmetros envolvidos.

c) Comandos de movimentação de áreas

Existe uma série de comandos cuja finalidade é ajeitar o layout através da cópia ou movimentação de trechos. Entre estes comandos, temos espelhamento, rotação, cópia, repetição (matriciação), ajuste, zoom. Estes comandos são implementados muito facilmente através do manuseio direto da memória de tela.

d) Visibilidade

Uma das operações mais importantes é a visibilidade, ou seja, ver-se somente algumas camadas do chip. Isto é obtido através de um mascaramento feito por hardware no gerador de cores, permitindo um esforço de software mínimo.

e) pegar e guardar um layout no disquete

Os layouts são gravados em CIF e podem ser trazidos para a posição do cursor, rodados ou espelhados. Da mesma forma, também podem ser jogados em disco em CIF.

f) DRC online

O EDMOS está conectado diretamente ao verificador de regras de projeto (DRC) projetado por (Teles-83), o que permite a verificação imediata dos layouts (ou parte deles) durante o processo de edição. Neste caso são mostrados piscando na tela os trechos os quais apresentam algum tipo de violação geométrica, como descrito em (Mead & Conway - 80).

O EDMOS prevê saídas para impressora alfanumérica e gráfica. No caso de impressora alfanumérica, cada pixel é representado pela sobreimpressão de 5 caracteres, representando cada uma das camadas NMOS. Os seguintes caracteres foram utilizados por darem uma boa sobreimpressão: O (difusão) / (polissilício) V (metal) . (implantação iônica) W (contato).

A impressão gráfica pode ser feita em uma impressora de pontos barata (Globus-M100) , que possui facilidade de se especificar diretamente as configurações das agulhas na linha (modo gráfico). A impressora tem uma pequena diferença entre as distâncias X e Y, mas no caso de checkplot isso não faz muita diferença.

A codificação para a impressora gráfica é feita através de uma matriz de 4 x 4, utilizando uma padronização utilizada na XEROX PARC. Isto permite a impressão numa página de um trecho com largura 198 lambda. A impressão de uma área com 198 x 198 lambda demora cerca de 5 minutos.

A vantagem do uso de impressora gráfica é grande. A rapidez e a relativa qualidade do desenho obtido permitem que se evite tirar saídas em plotter, que apresenta uma certa dificuldade de visualização (Borges-83) e um tempo enorme de desenho. O ideal, seria, entretanto, uma impressora gráfica colorida, mas esse é um equipamento difícil de obter e muito caro.

FUNCIONAMENTO E PRINCIPAIS ALGORITMOS DO EDMOS

O EDMOS trabalha preenchendo áreas da tela com cores, e consultando o conteúdo da tela. Isso, na tecnologia utilizada (raster scan) é extremamente simples, pois basta consultar diretamente a memória da tela. Para aumentar a portabilidade do software, criamos um conjunto de funçoes que implementam essas primitivas de acesso:

ESCVID (x,y,cor) : colore uma posição da tela

LEVID (x,y,cor) : informa a cor de um ponto da tela

RETANG (x,y,dx,dy,cor,opção) : desenha um retângulo colorido, superpondo (fazendo "ou" dos bits) ou reescrevendo os pontos

MASCOR (masc) : seleciona a máscara de visibilidade (video lookup table).

Os algoritmos mais interessantes são o que desenha o cursor e o que extrai o desenho da tela, transformando-o para CIF.

O cursor é um retângulo branco, que pode ser movido na tela. Não existe no nosso equipamento cursor gerado por hardware. Para implementar esta função, ao colocar o cursor na tela, copiamos previamente o conteúdo das posições da tela onde ele será colocado para uma área auxiliar e substituímos estes pontos pela cor branca. Para apagar o cursor recolocamos os pontos a partir da área auxiliar. A movimentação do cursor é composta de um apagamento seguido de um redesenho do cursor. Sempre que alguma função vai ser executada, o cursor é previamente apagado, e terminada a função, é recolocado.

A extração do desenho é feita varrendo-se a área delimitada por cursor, linha a linha, de baixo para cima, localizando-se as sequências de pontos com a mesma cor. É mantida uma lista com as sequências da linha anterior. As novas sequências são comparadas com as da lista, procurando-se achar sequências idênticas. Vai-se acumulando o tamanho(no sentido vertical) das sequências idênticas, ou seja, vai-se obtendo o tamanho dos re - tângulos. As sequências da linha atual sem equivalência na lista, são inseridas (pois são na verdade novos retângulos). As sequências da lista, sem equivalência na sequência da linha atual, são consideradas como fim de retângulo, e retiradas (ou seja, é da da sua saída em CIF).

DA NECESSIDADE DE UM EDITOR HIERÁRQUICO

À primeira vista, o projeto de um circuito integrado pode ser comparado ao projeto top down de software. Tem-se, a partir da idéia geral do problema, a divisão do projeto em blocos, a confecção da planta baixa (que é a divisão em partes do espaço físico), e o projeto de cada célula, que pode conter subcélulas, e assim por diante. Podem existir células pré-programadas (standard cells, bibliotecas de células), com entradas e saídas bem definidas que são utilizadas por todos os projetistas. O projeto das células é feito às vezes como caixas vazias em que existem apenas definidos os pontos de conexão,em analogia com subrotinas "dummy", que possuam especificados apenas seus parâmetros formais.

Infelizmente, uma abordagem puramente top-down é utópica neste caso. A alocação do espaço físico é uma outra dimensão que ultrapassa a noção algorítmica do projeto. Existem neste caso problemas geométricos que tornam muito complexo o projeto: o encaixe das células, sua interligação, minimização do layout, alimentação elétrica, etc...

Daí, dividimos o problema em duas partes: projeto das células e montagem das células . Essas duas partes se interrelacionam e criam um círculo vicioso: o projeto depende das especificações para a montagem e a montagem depende do projeto.

Essa situação é sempre minimizada quando se utilizam células projetadas (por exemplo , obtidas em bibliotecas de células) ou geradas por geradores automáticos como descritos em (Teles & Schmitz-85). O que acaba contando, é a experiência anterior do projetista, que se torna capaz de estimar com relativa precisão o tamanho das células e arbitrar as disposições de montagem que dêem como resultado um layout mais compacto. Entretanto, este é um problema que permanece, e para o qual não existe ainda uma solução definitiva.

Para auxiliar a parte de montagem, é fundamental um editor que consiga ajudar no pro - cesso da alocação do espaço e ligação das partes componentes do circuito. Este editor deve ter a noção de hierarquia, ou seja, ser capaz de tratar da colocação de células em disposição umas dentro das outras, recursivamente, para que a montagem e projeto tenham uma interface menos complicada.

VISÃO GERAL DO EDCI

O EDCI foi projetado para poder rodar tanto no computador principal,utilizando a esta-

ção de trabalho apenas como unidade de saída, ou executar mesmo na própria estação de trabalho, de maneira stand-alone. Essa independência é obtida pela utilização de rotinas gráficas modulares e pelo fato do editor ser escrito em "C" bastante padrão.

Da mesma forma que o EDMOS, ele usa uma tela alfanumérica e uma tela gráfica. Na tela alfanumérica se mantém todo o diálogo com o operador, e são mostrados todos os status da edição (posição do cursor, nome do símbolo e do layout, etc...). Na tela gráfica, aparece o layout que se está projetando, numa escala arbitrária. Essa separação de telas possibilitou o uso de terminais gráficos com resolução bastante modesta (240 x 400 pontos).

No EDCI, o chip é tratado como um caixote visto do alto. Dentro deste caixote podem existir três tipos de coisas: retângulos coloridos (box de CIF), textos ou outros caixotes (símbolos mais internos, células de CIF). Dentro dos caixotes internos podem existir outros retângulos, textos e caixotes, e assim sucessivamente.

Foi minimizada a quantidade de coisas que o editor desenha automaticamente. Como agora estamos tratando de layouts completos, o tempo de desenho pode ser muito grande. É melhor, então, que seja opção do operador, por exemplo, mostrar um símbolo apenas através de seu contorno e conexões, do que todo o interior expandido. Além disso, é permitido o uso de escala menor que 1, ou seja, pode-se ter uma visão macroscópica do chip e nesse caso o numero de elementos desenhados na tela é enorme, caso se deseje ver o conteúdo de todos os subsímbolos.

OPERAÇÃO DO EDCI

Inicialmente é mostrado na tela o contorno de um retângulo, representando o chip, e dentro dele, outros retângulos vazios, representando o nível imediatamente inferior de células (ou seja as células chamadas pelo nível mais global), e os retângulos (Boxes CIF) coloridos deste nível.

Existe um cursor retangular que o operador pode mover de forma semelhante ao EDMOS para se posicionar dentro do layout. Eventualmente, o operador vai querer entrar num subsímbolo, alterar seu conteudo, entrar num subsímbolo ainda mais interno, sair dele, e assim por diante, passeando de uma forma relativamente confortável dentro do chip. É possível isolar um trecho do layout ou um símbolo para edição em escala maior, alteração de tamanho dos símbolos ou seu posicionamento, inclusão, criação e retirada de elementos no circuito, etc... O controle da posição de navegação dentro do chip é feita completamente de maneira visual, ou seja, através do cursor. Para manter-se a noção de hierarquia, ao sair de um símbolo (opcionalmente) pode-se mandar recobri-lo, e assim tornar a visualização mais simples.

Os comandos do EDCI implementados em sua primeira versão são os seguintes:

- movimenta cursor
- cria, remove, entra e sai de símbolo
- cria e remove box

. ajusta tamanhos e posições de símbolos
. expande chamadas de simbolos até qualquer nível
. isola, amplia e reintegra símbolos
. cria matrizes de símbolos
. salva e recupera o conteúdo do cursor
. cria e remove hierarquias
. traz simbolo em CIF para o banco de dados ou dele extrai um arquivo em CIF
. mostra, altera, manuseia tabela de símbolos
. seleciona janela de edição, visibilidade, escala
. comando de ajuda online para todos os comandos
. possui interface para conexão com o DRC e o gerador de rotas automático.

ALGORITMOS

Os algoritmos deste editor e suas principais estruturas de dados são baseados em uma árvore que define as relações de hierarquia e o conteúdo das células. No manuseio desta árvore, são utilizados algoritmos conhecidos de busca, manuseio, inclusão e exclusão em árvores aplicadas a processamento gráfico, como descritas em (Newman & Sproull), bem como toda a parte de manuseio de transformações matriciais descritas naquela referência.

Os principais algoritmos podem ser achados em (Pires & Borges-85).

CONCLUSÃO

O EDMOS e o EDCI estão em funcionamento no NCE-UFRJ e são utilizados em cursos de graduação e pós-graduação de projeto VLSI. São utilizados ainda pela equipe de pesquisa em projeto VLSI do NCE (Silva Fº, Schmitz, Faller -83), e foram importantes ferramentas no projeto de um dos primeiros chips que foram fabricados no Brasil.

A versão atual destes editores é destinada à tecnologia NMOS, mas eles foram construídos de forma paramétrica, para permitir um mínimo esforço de conversão para outras tecnologias (por exemplo, CMOS ou HMOS). São relativamente portáteis, e rodam com um mínimo de necessidades computacionais.

A parte gráfica foi projetada especialmente para poder trabalhar com qualquer tamanho de tela e o protocolo gráfico está centralizado em uma única rotina para possibilitar seu transporte para virtualmente qualquer terminal gráfico Raster scan colorido.

1 . BORGES, J.A.S. Ferramentas gráficas para o Projeto de Circuitos Integrados
Anais do Congresso Nacional de Informática - 1983

2 - LEWIS, D. Layout design on an alphanumeric terminal
VLSI Design, vol. II, no. 4 (1981)

3 - FAIRBAIRN, D et ROWSON, J. ICARUS: an interactive integrated circuit layout program anais da 15a. Design Automation Conference, 1978

4 - NEWMAN, W. et SPROULL Principles of interactive computer graphics
2nd. edition - MacGraw Hill - 1979

5 - PIRES, O.V.S. et BORGES, J.A.S. Estruturas de dados e algoritmos de um editor hierárquico para projeto VLSI - submetido à comissão de seleção do II SBCCI - 1985

6 - OUSTERHOUT, J. CAESAR: an interactive editor for VLSI layouts
VLSI Design, vol. II, no. 3 (1981)

7 - TELLES, A.A.S. Verificação de Projeto de Circuitos Integrados
Tese de mestrado - COPPE-UFRJ - 1983

8 - SILVA FQ, Y., SCHMITZ, E., et FALLER, N. É possivel projetar circuitos integrados no Brasil ? Anais do Congresso Nacional de Informática - 1983

9 - TELLES, A.A.S et SCHMITZ MAKEROM - um gerador automático de ROM
Submetido à comissão de seleção do II SBCCI - 1985

ESTRUTURAS DE DADOS E ALGORITMOS DE UM EDITOR GRAFICO PARA PROJETOS VLSI

J. A. S. BORGES*

O. V. S. PIRES**

RESUMO

ESTE TRABALHO TEM POR FINALIDADE DESCREVER ESTRUTURAS DE DADOS E ALGORITMOS UTILIZADOS POR UM EDITOR GRÁFICO HIERÁRQUICO PARA PROJETOS DE CIRCUITOS INTEGRADOS VLSI, DESENVOLVIDO NO NCE/UFRJ.

ABSTRACT

THIS WORK INTENDS TO DESCRIBE DATA STRUCTURE AND ALGORITHMS USED IN A HIERARCHICAL GRAPHIC EDITOR FOR INTEGRATED VLSI CIRCUITS DESIGN, DEVELOPED AT NCE/UFRJ.

* INFORMATICO (1980 - UFRJ); PROCESSAMENTO GRÁFICO; ESTAÇÕES DE CAD; UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, NÚCLEO DE COMPUTAÇÃO ELETRÔNICA , CAIXA POSTAL 2324, RIO DE JANEIRO, CEP 20001.

** GRADUANDO EM INFORMÁTICA (UFRJ); SOFTWARE BÁSICO; SISTEMAS OPERACIONAIS COMPILADORES; CAD PARA VLSI; UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, NÚCLEO DE COMPUTAÇÃO ELETRÔNICA, CAIXA POSTAL 2324, RIO DE JANEIRO, CEP 20001.

1. INTRODUÇÃO

Este trabalho descreve o funcionamento de um editor gráfico hierárquico para projetos de circuitos integrados VLSI, conforme especificado em (Borges-85).

Durante a concepção deste editor, tivemos como objetivo principal a fácil interação com usuário; achamos interessante que o EDCI pudesse dialogar com o projetista de maneira essencialmente gráfica. Para isto, ele se utiliza de duas telas: uma gráfica, onde se desenham as porções solicitadas do lay-out, e uma tela alfanumérica, onde são mostradas diversas informações sobre o estado atual da edição. Na tela gráfica, os elementos do circuito são representados por uma codificação de cores, correspondendo cada cor a uma camada específica utilizada na tecnologia de fabricação. As subcélulas são representadas por seus contornos, de modo a não sobrecarregar a imagem manipulada.

As funções que o EDCI é capaz de realizar podem ser divididas em duas categorias: funções que atuam sobre a hierarquia da montagem de células e funções que atuam sobre os elementos básicos do circuito. No primeiro caso, é possível verificar os conteúdos das células, alterar suas dimensões e posicionamento, removê-las, entre outras; a segunda categoria permite, entre outras, criar, remover e ajustar as posições dos componentes.

O EDCI permite que o projetista se posicione em um trecho qualquer do lay-out através de um cursor retangular representado por um contorno branco cujos movimentos básicos (deslocamento e ampliação) são comandados pelo operador através do teclado. A movimentação pode ser feita em modo lento ou rápido, possibilitando o percurso de todo um lay-out em tempo reduzido (Borges-83).

Para permitir o funcionamento do editor em máquinas de pequeno porte, tivemos que elaborar uma base de dados em memória secundária que conseguisse manter atualizadas todas as modificações incluídas durante o processo de edição, conservando na memória principal apenas as informações relativas a porção do lay-out sendo editada (usualmente, uma célula). Calcado nesta forma intermediária de armazenamento e sobre um mecanismo de transformações matriciais, o EDCI é muito eficiente e versátil no tratamento das informações gráficas.

2. DESCRIÇÃO DO BANCO DE DADOS

2.1. CIF e o formato DB/SYM

A linguagem CIF (Caltech Intermediate Format) é uma das linguagens mais utilizadas para descrição da geometria de Circuitos Integrados. Ela permite especificar para cada célula definida a combinação e posicionamento das diversas máscaras empregadas no processo de fabricação; a esses elementos chamamos boxes. Também é possível especificar subcélulas contidas em uma determinada célula, o que chamamos call. Ainda pa

ra efeito de extração e simulação de circuito, é interessante que se possa atribuir um nome a uma determinada posição dentro de uma célula, como descrito em (Telles & Souza-85); a estes pontos chamamos nós.

Para se editar um lay-out com o EDCI a partir de sua descrição em CIF, foi necessária a construção de um programa pré-processador que convertesse esta descrição para um formato fixo mais facilmente manuseável para o editor, já que o texto livre não apresenta essas características.

Basicamente este conversor identifica os símbolos (células) definidos no texto fonte e gera dois arquivos que conterão as informações geométricas necessárias à edição do lay-out. Estes arquivos foram denominados DB e SYM.

Nosso pré-processador é capaz de identificar estes elementos e compactá-los de forma inteligível ao EDCI numa estrutura de dados que examinaremos a seguir.

2.2. Organização em disco

Para cada símbolo definido no texto fonte em CIF é gerado um registro correspondente no arquivo DB e um conjunto de registros no arquivo SYM.

No arquivo DB estão presentes as seguintes informações:

- Número e nome do símbolo;
- Dimensões do símbolo (bounding-box):
  - coordenadas do canto inferior esquerdo;
  - comprimento e altura do símbolo;
- Apontadores para a descrição do símbolo no arquivo SYM.

O registro inicial do arquivo DB é um cabeçalho contendo basicamente o número de símbolos guardados naquele banco de dados, a tecnologia de fabricação e o número de setores ocupados do arquivo SYM.

Neste último arquivo, distinguem-se quatro tipos de registros, cada um deles descrevendo um tipo de componente da célula, a saber:

- boxes:
  - camada (layer) do box;
  - coordenadas do centro;
  - comprimento e altura;
- calls:
  - número do símbolo chamado;
  - matriz de transformação da chamada;
- nós:
  - coordenadas dos nós;
  - tipo e camada do nó, conforme (Telles & Souza-85);

- nome do nó;

- texto de comentário.

O acesso à descrição do símbolo no arquivo SYM é feito através dos ponteiros existentes no arquivo DB. Eventualmente, alguns registros podem não existir, por exemplo, se o símbolo não contiver subsímbolos. Neste caso uma informação inválida é colocada no ponteiro correspondente.

2.3. Organização em memória

A estrutura de dados em memória principal compõe-se de cinco tabelas:

- tabela de símbolos:
  Esta tabela é preenchida no início da edição com o conteúdo do arquivo DB;

- tabela de boxes:
  É preenchida com a descrição dos boxes da célula sendo correntemente editada;

- tabela de calls:
  Contém as informações sobre as chamadas a símbolos contidas na célula-corrente;

- tabela de nós:
  É preenchida com a descrição dos nós da célula-corrente;

- vetor de texto:
  Contém o texto de comentário do símbolo corrente.

A tabela de símbolos permanece em memória durante toda a edição, só sendo regravada em disco no final do processo. Passo a passo, conforme os símbolos vão sendo editados, as informações das quatro tabelas restantes vão sendo trazidas e regravadas em disco, como apêndices do arquivo SYM. Para que fosse atenuado ao máximo o tempo dispendido em entrada e saída para preenchimento das tabelas, só regravamos um símbolo em disco se houver sido produzida alguma alteração em sua estrutura durante a edição.

3. INTERFACE ENTRE IMAGEM E BANCO DE DADOS

3.1. Representação de um símbolo na tela

Conforme já foi mencionado, as informações contidas na descrição de um símbolo que possuem correspondente gráfico são boxes e calls. Aqueles são representados por áreas retangulares coloridas e estes por seus contornos (bounding-boxes) desenhados em azul-claro.

A versão atual do EDCI está dedicada à tecnologia nMOS, existindo portanto uma cor associada a cada uma das cinco camadas existentes, conforme (Borges-83). Estas cores podem ser combinadas dando a impressão de superposição de boxes na tela gráfica.

3.2. Associação entre o Banco de Dados e a Tela

Ao símbolo correntemente sendo editado associamos uma transformação matricial que permite fazer correspondência entre o sistema de coordenadas do símbolo e o sistema de coordenadas de tela gráfica. À medida que se caminha em profundidade na hierarquia dos símbolos, as transformações vão sendo compostas de modo a refletirem a correspondência atual entre desenho/tela.

Assim como determinada transformação permite fazer correspondência entre as coordenadas do símbolo e as coordenadas da tela, o contrário é conseguido através da aplicação da inversa da transformação, que mapeia pontos da tela em pontos do símbolo.

3.3. Matrizes de Transformação

O formalismo descrito a seguir encontra-se em (Newman & Sproull-79).

As transformações associadas a símbolos são representadas por matrizes 3 x 3, sendo possível através delas: transladar um símbolo, espelhá-lo em relação ao eixo X ou Y e girá-lo de 90, 180 ou 270 graus.

Matematicamente, seja P = (x,y) um ponto do plano. Vamos representá-lo através da matriz:

$$[\, x \;\; y \;\; 1 \,]$$

O ponto resultante da aplicação de uma transformação representada pela matriz T ao ponto P é dado por P x T.

Assim, se quisermos transladar a origem do sistema para o ponto (u, v), a matriz de transformação correspondente será:

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ u & v & 1 \end{vmatrix}$$

e o produto P x T valerá

$$[x+u \;\; y+v \;\; 1]$$

que representa o ponto $(x+u, y+v)$.

Para espelhamento em relação a Y, X e rotação de t graus (t = 90, 180 ou 270 para geometria Manhattan), as matrizes são as seguintes:

$$\begin{vmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad \begin{vmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad \begin{vmatrix} \cos(t) & -\text{sen}(t) & 0 \\ \text{sen}(t) & \cos(t) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Composições de transformações são obtidas multiplicando-se as matrizes que as representam. Desta forma, se a um símbolo está associada a transformação T e este símbolo chama outro com transformação U, a transformação associada ao segundo quando

ele for editado será U x T.

Na implementação, consideramos o fato de que a terceira coluna das matrizes valeria sempre 0 0 1, o que nos permitiu representá-las por um vetor de 6 elementos, reduzindo consideravelmente o tempo de acesso a cada um deles.

3.4. Stack de chamadas

Via de regra, a edição no EDCI começa sempre do símbolo fundamental, identificado no banco de dados como sendo o de número 0. A este símbolo associa-se uma transformação tal que as coordenadas de seu canto mínimo venham a coincidir exatamente com as coordenadas (0,0) da tela. Se (xmin, ymin) são as coordenadas do canto mínimo, não é difícil ver que a transformação será:

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -xmin & -ymin & 1 \end{vmatrix}$$

Uma das funções do EDCI (chamada Entra em Símbolo) permite, estando em um determinado nível, passar a editar uma subcélula chamada pelo símbolo corrente. Terminada a edição da subcélula, o projetista pode querer retornar ao símbolo de nível imediatamente anterior. Estes fatos sugerem que as informações sejam empilhadas durante o transcurso da hierarquia. Em geral, empilhamos o seguinte:

- número do símbolo corrente;
- transformação atual;
- limites de movimentação do cursor;
- escala atual de edição;
- posição atual do cursor.

Estas informações são suficientes para retornar um nível na hierarquia quando assim for desejado.

Outras funções que manipulam a hierarquia estão descritas em (Borges-85).

4. ALGORITMOS MAIS IMPORTANTES

O processo de edição torna-se suficientemente complexo na medida em que as imagens gráficas devem acompanhar as alterações realizadas sobre a estrutura de dados que representa a célula; isto implica em redesenhos constantes da tela gráfica, para que o desenho reflita a realidade das estruturas de dados.

Das diversas funções implementadas no EDCI, selecionamos algumas cujos algoritmos apresentam aspectos interessantes tanto do ponto de vista gráfico quanto algoritmico. Eles foram calcados sobre uma base de primitivas cuidadosamente elaboradas para permitir eficiência nas operações solicitadas. Dentre estas primitivas, podemos citar:

- preencher as tabelas de boxes, calls, nós e texto, o que corresponde a trazer as informações do disco;
- regravar estas informações em disco quando necessário;
- empilhar e desempilhar informações sobre o contexto atual de edição;
- aplicar a transformação atual a um ponto;
- aplicar a transformação atual a um box;
- multiplicar e inverter matrizes;
- desenhar um box na tela;
- desenhar um contorno na tela;
- apagar uma região da tela.

4.1. Desenho de um símbolo na tela

Esta função é chamada no EDCI toda vez que um símbolo deva ser desenhado na tela submetido e uma determinada transformação.

```
Procedimento MostraSímbolo (S, T)
S - Número do Símbolo
T - Matriz de Transformação
Início
    Traz o símbolo S do disco
    Para cada box do símbolo
        Aplica T ao box, obtendo (bx, by, bdx, bdy)
        Desenha (bx, by, bdx, bdy) na tela
    Para cada call do símbolo
        Seja M a matriz associada à chamada
        Seja N o símbolo chamado
        U = M x T
        Aplica U ao bounding-box de N, obtendo (bx, by, bdx, bdy)
        Desenha o contorno (bx, by, bdx, bdy)
Fim.
```

4.2. Entrada em Símbolo

Esta operação consiste em fazer com que o processo de edição passe a se realizar sobre uma subcélula do símbolo atual, o que corresponde a descer um nível na hierarquia.

```
Procedimento EntraSímbolo
Início
    Seja T a transformação atual
    Seja S o símbolo que está sob o cursor
    Seja M a matriz de chamada do símbolo S
    Se o símbolo atual foi alterado
```

```
        Guarda-o em disco
    Empilha o contexto atual
    U = M x T
    Aplica U ao bounding-box de S, obtendo (bx, by, bdx, bdy)
    Apaga a região (bx, by, bdx, bdy)
    Recalcula limites de movimentação do cursor
    MostraSímbolo (S, U)
    T = U
    Posiciona o cursor no canto inferior esquerdo do símbolo S
Fim.
```

4.3. Saída de Símbolo

É o processo inverso ao de entrada em símbolo. Acrescentamos a facilidade de o conteúdo do símbolo abandonado continuar ou não a ser visível.

```
Procedimento SaiDeSímbolo
Início
    Pergunta ao Usuário: "Quer recobrir o símbolo ?".
    Guarda a resposta em resp.
    Se o símbolo atual foi alterado
        Guarda-o em disco.
    Desempilha o contexto de edição.
    Se resp foi afirmativa
        Apaga a região que estava sendo editada.
    Redesenha (S, T, região anterior).
Fim.
```

4.4. Redesenho de uma região

O processo de atualização da tela gráfica muitas vezes se reflete em apagar um trecho que tenha sido alterado e redesenhar somente aquele trecho do símbolo. Este procedimento difere de MostraSímbolo apenas no fato de que somente a porção indicada é redesenhada.

```
Procedimento Redesenha (S, T, R)
S - Número do símbolo.
T - Matriz de Transformação.
R - Retângulo (x, y, dx, dy) delimitando o trecho a redesenhar.
Início
    Traz o símbolo S do disco.
    Para cada box do símbolo
        Aplica T ao box, obtendo (bx, by, bdx, bdy).
        Se (x, y, dx, dy) intercepta (bx, by, bdx, bdy)
```

```
            Desenha o box (bx, by, bdx, bdy).
    Para cada call do símbolo
        Seja M a matriz da chamada.
        Seja N o símbolo chamado.
        U = M x T.
        Aplica U ao bounding-box de N, obtendo (bx, by, bdx, bdy).
        Se (x, y, dx, dy) intercepta (bx, by, bdx, bdy)
            Desenha o contorno (bx, by, bdx, bdy).
Fim.
```

4.5. Expansão de Símbolo

A operação de expansão no EDCI consiste em visualizar um símbolo e seus subsímbolos recursivamente.

A expansão é utilizada principalmente quando se deseja visualizar todo o layout de um chip sendo editado, por exemplo.

```
Procedimento Expande (S, T)
S - Número do símbolo a expandir.
T - Matriz de Transformação.
Início
    Traz o símbolo S do disco.
    Para cada box do símbolo
        Aplica T ao box, obtendo (bx, by, bdx, bdy).
        Desenha o box (bx, by, bdx, bdy).
    Para cada call do símbolo
        Seja N o símbolo chamado.
        Seja M a matriz da chamada.
        U = M x T.
        Expande (N, U).
Fim.
```

4.6. Apagamento de boxes

A função de apagamento de boxes permite remover todos os boxes que estejam situados sob a região delimitada pelo cursor. A complexidade deste algoritmo reside no fato de que, ao se apagar uma porção de um box, podem surgir de zero a quatro novos boxes distintos.

```
Procedimento ApagaBox (C)
C - Conjunto de cores a apagar.
Início
    Seja T a transformação atual e I a sua inversa.
    Seja S o símbolo atual.
```

```
Seja R = (xcur, ycur, dxcur, dycur) as coordenadas do cursor.
Aplica I ao cursor obtendo (x, y, dx, dy).
Para cada box do símbolo atual
    Seja L o layer do box.
    Se Cor (L) pertence a C
        Sejam (xb, yb, dxb, dyb) as coordenadas do box.
        Se (x, y, dx, dy) intercepta (xb, yb, dxb, dyb)
            Remove o box da tabela.
            Acrescenta os boxes complementares.
Apaga a região do cursor R.
Redesenha (S, T, R).
Fim.
```

4.7. Seleção de Janela

É possível durante o processo de edição que se queira visualizar regiões do lay-out em escala ampliada ou mesmo visualizar o lay-out de todo um chip. Tendo em vista as limitações das dimensões da tela, torna-se necessária uma função que possibilite selecionar o trecho do lay-out a ser mostrado. Isto se reflete numa alteração da matriz de transformação corrente, sendo conseguido pelo algoritmo abaixo:

```
Procedimento SelecionaJanela (X, Y)
(X, Y) - Ponto a partir do qual o lay-out será mostrado.
Início
    Seja T a transformação atual.
    Seja U a transformação que translada a origem para (X, Y)
```

$$U = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X & -Y & 1 \end{vmatrix}$$

```
    T = T x U.
Fim.
```

5. CONCLUSÃO

A versão atual do EDCI, já em funcionamento no NCE desde o início de 1985, foi programada em linguagem C para executar em um mini-computador PDP 11/70 com sistema operacional Unix.

Quanto a tamanho e desempenho, o EDCI é um programa de aproximadamente 5.000 linhas que necessita de 128 Kbytes de memória para execução. Seu rápido tempo de resposta a comandos, mesmo com alto índice de utilização da máquina, deve-se principalmente ao fato de as rotinas básicas largamente solicitadas pelo editor haverem sido elaboradas de maneira a funcionarem o mais eficientemente possível, evitando-se pontos de estrangulamento no fluxo normal de execução. Além disso, foram utilizados re-

cursos de bufferização promovidos pelo Unix para entrada e saída de dados que permitiram tornar quase instantâneo o processo de atualização de imagens.

Deve ser enfatizada a questão da portabilidade: por haver sido programado em C de forma paramétrica e modular, o EDCI é um programa que pode ser facilmente adaptado para outros equipamentos, eventualmente micro-computadores, com um mínimo de alterações.

6. BIBLIOGRAFIA

Borges, J. A. S. - Ferramentas Gráficas para o Projeto de Circuitos Integrados - Anais do Congresso Nacional de Informática, 1983.

Borges, J. A. S. - Editores Gráficos para Projetos de Circuitos Integrados - Submetido à Comissão de Seleção do II Simpósio Brasileiro de Concepção de Circuitos integrados.

Teles, A. A. S. et Souza, L. F. P. - EXTRAI - Extrator de Circuitos Integrados NMOS - Submetido à comissão de seleção do II Simpósio Brasileiro da Concepção de Circuitos Integrados.

Newman, W. et Sproull, R. - Principles of Interactive Computer Graphics - 2nd. edition - Mac Graw Hill - 1979

ALLENDE: UMA LINGUAGEM DE ESPECIFICAÇAO DE LAYOUTS DE CIRCUITOS INTEGRADOS

José Monteiro da Mata*

Sumário

ALLENDE é uma linguagem simples e poderosa para a especificação de layouts de circuitos integrados. Em ALLENDE o layout é descrito hierarquicamente como uma composição de células; posições ou tamanhos absolutos nunca são especificados. A descrição do layout é traduzida para equações lineares, que expressam regras de desenho e posições relativas dos elementos do layout. A solução dessas equações determina o layout absoluto, que garantidamente não contem violações das regras de desenho.

ALLENDE consiste de 5 subrotinas que podem ser chamadas de um programa em Pascal ou C, e foi implementado para a tecnologia nMOS.

Abstract

ALLENDE is a simple and powerful procedural language for VLSI layout. In ALLENDE the layout is described hierarchically as a composition of cells; absolute sizes or positions are never specified. The layout description is translated into linear constraints, which express design rules and relative position of the layout elements. By solving these constraints we obtain the absolute layout, which is guaranteed to be free of design rule violations.

ALLENDE consists of five procedures to be called from a Pascal or C program, and has been implemented for the nMOS technology.

* Engenheiro eletricista (UFMG, 1974), Ph.D. em ciência da computação (Princeton, 1984); CAD, software básico; Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Caixa Postal 702, Belo Horizonte, MG 30000

1. Introdução

Uma revisão das técnicas de projeto de circuito integrados torna-se necessária, devido ao custo associado ao projeto de circuitos complexos, à necessidade de tornar a tarefa de projeto de circuitos acessível a um maior número de pessoas, e à necessidade de ferramentas mais poderosas para lidar com modificações no projeto. São necessários métodos que aumentem a produtividade do projetista.

O desenho do layout é uma etapa crítica no projeto de circuitos integrados, porque ela envolve ferramentas dispendiosas e uma grande quantidade de intervenção humana, e também devido a seus efeitos nos custos de produção. Um dos objetivos das ferramentas de layout é produzir layouts corretos, mas ainda hoje há a proliferação de programas verificadores de layout, como programas verificadores de regras de desenho.

Neste trabalho nos concentramos no problema da produção de layouts. Temos dois objetivos principais em mente:

- possuir uma ferramenta poderosa que permita ao projetista obter facilmente um layout correto para seu projeto;
- possuir um componente que possa ser expandido e integrado com outros componentes de um sistema de projeto associado a uma metodologia estruturada de projetos.

Nossa abordagem para o problema de layouts de circuitos integrados é usar uma linguagem para descrever hierarquicamente a estrutura do circuito e a topologia do layout, e usar equações lineares para representar internamente o layout.

Nosso sistema de layout, ALLENDE[5], sucessor de ALI[3][7], possui características que o diferenciam das ferramentas de layout existentes. O uso de uma linguagem "procedural" e a representação do layout através de equações distingue ALLENDE de todos os editores gráficos de layout e da maioria das linguagens para descrição de layouts. A diferença com outras linguagens "procedurais" que usam equações para representar layouts consiste na forma de descrição do layout, no tipo de equações utilizadas, e também na forma de implementação do sistema. O resultado é a capacidade, flexibilidade e eficiência alcançados por ALLENDE.

2.1. Idéias básicas

Nossa abordagem para o problema de obtenção de layouts de circuitos integrados é através do uso de uma linguagem para descrição da estrutura do layout. A partir da representação textual do layout, equações lineares são geradas, resolvidas, e então o layout físico é obtido. As características da linguagem dependem, obviamente, da classe de objetos manipulados e do tipo de manipulação.

Em nosso sistema, o único objeto que temos são células. Uma célula corresponde a um retângulo com estrutura interna e terminais de comunicação.

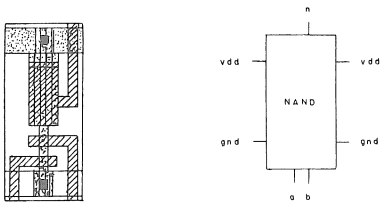


Fig. 1 - Uma célula nand

A composição de duas células é feita especificando sua posição relativa (esquerda, direita, acima, abaixo), e o resultado é outra célula. Uma célula individual pode também sofrer rotação ou espelhamento.

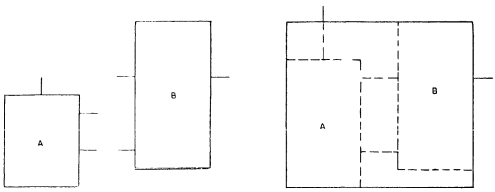
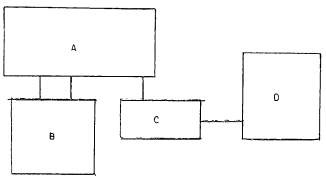


Fig. 2 - Composição de células

Na composição de células não é necessário se preocupar com o "alinhamento" dos fios de comunicação entre células; as condições para alinhamento são expressas nas equações geradas, e então o tamanho de uma célula depende do contexto no qual ela é instanciada.

A primeira idéia básica é descrever o layout hierarquicamente como a composição de células. No último nível da hierarquia (folhas da árvore) estarão **células do sistema** (contatos, transistores, etc.) ou células **rígidas** (partes do layout previamente definidos).



(A above (B left C)) left (rotated90 D)

Fig. 3 - Descrição estruturada do layout

Ao usar uma célula do sistema devemos especificar os fios em cada lado da célula. Quando é feita a composição de duas células, os fios a serem conectados, e os fios de comunicação da célula resultante, são determinados por contexto.

A segunda idéia básica é usar uma linguagem intermediária para descrever a estrutura do layout. Esta estrutura representa a posição relativa das células, em uma forma estruturada como na figura 3. O programa do usuário gera esta forma intermediária, a partir da qual equações lineares são extraídas, e então obtido o layout final. O objetivo é separar aspectos da linguagem do usuário (representação e processamento) de aspectos do layout (geração de equações e produção do layout); a forma intermediária lida somente com a estrutura do layout, enquanto a linguagem do usuário pode ter todo o poderio de uma linguagem "procedural".

Nosso sistema de layout é baseado nestas idéias. Existe uma linguagem do usuário, ALLENDE, que é nada mais do que Pascal ou C com a adição de algumas subrotinas e funções. A saída do programa do usuário é a estrutura do layout em linguagem intermediária (PIF), da qual são extraídas as equações lineares, que são resolvidas, produzindo o layout absoluto em CIF (Caltech Intermediate Form).

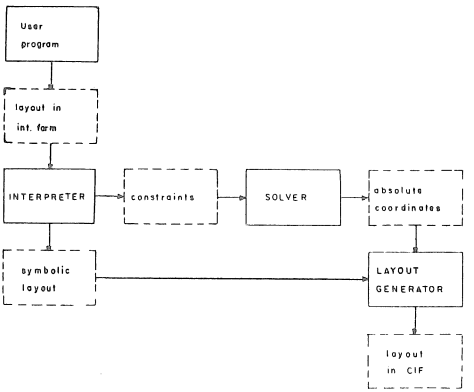


Fig. 4 - Processo de geração do layout

Erros podem ocorrer durante compilação do programa do usuário, execução, interpretação do programa PIF, e solução do sistema de equações. Erros de compilação e execução são os usuais, detetados pelo compilador Pascal ou C ou durante execução. No caso de erro durante interpretação da forma intermediária que descreve o layout, o interpretador PIF identifica a célula onde ocorreu o erro. O único erro possível durante o processo de solução das equações é uma célula rígida não cabendo no contexto onde ela é instanciada; nesse caso o programa que resolve as equações aponta a célula que causou o erro.

2.2. A linguagem ALLENDE

ALLENDE (A Layout Language Effective for nMOS Design) é um conjunto de subrotinas e funções que podem ser chamadas de um programa em Pascal ou C, permitindo ao usuário descrever o layout de um circuito VLSI. Basicamente, o usuário descreve células e seu posicionamento relativo. A hierarquização de células aparece naturalmente através do uso de subrotinas para descrever células.

O usuário pode fazer uso de todo o poderio do Pascal ou C. As subrotinas e funções para descrever o layout permitem uma enorme quantidade de parametrização, o que permite obter layouts completamente diferentes somente alterando um parâmetro no programa.

A saída do programa do usuário é o layout na forma intermediária (PIF), a partir da qual equações lineares são obtidas, resolvidas, e produzido o layout absoluto em CIF. O layout obtido garantidamente não contém violações das regras de desenho.

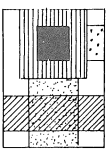
As seguintes subrotinas permitem ao usuário descrever um layout:

```
syscell(kind,wire1,wire2,wire3,wire4,ratio)
extcell(filename)
place(operator)
begincell(cellname)
endcell(wirenames)
```

syscell especifica uma célula do systema. **extcell** especifica uma célula externa. **begincell** e **endcell** são usadas para delimitar uma célula composta. **place** especifica o operador a ser aplicado para uma composição de células.

Como a ação destas subrotinas é gerar algum código intermediário a ser interpretado mais tarde, no programa do usuário podem existir comandos em Pascal ou C misturados com chamadas a estas subrotinas. O usuário pode também definir novas subrotinas em termos destas subrotinas básicas.

A figura 5 mostra um programa ALLENDE, em C, com o layout correspondente. Neste programa o uso da variável "clock" representa um exemplo de parametrização.



```
#include "/va/allende/usr/def.h"
main()
 {
  wiretype clock;
  clock = poly(2);
  begincell("example");
    syscell(CONTACT,nowire,nowire,diff(2),metal(3),0);
    place(ABOVE);
    syscell(CROSSING,clock,metal(3),clock,metal(3),0);
  endcell("  ");
 }
```

Fig. 5 - Um programa ALLENDE

A subrotina syscell permite a especificação de uma célula do sistema. CONTACT, TRANSISTOR e CROSSING são algumas das células do sistema. O único lugar onde o usuário deve especificar fios (wires) é nas células do sistema, onde devem ser especificados os fios nos lados esquerdo, superior, direito e inferior da célula. As funções metal(width), poly(width), diff(width), e a função mais geral wire(layer,width), permitem a especificação de um fio. Aqui é onde pode ser feita muita parametrização. "layer" e "width" podem ser parametrizados; além disso, um "wire" pode ser uma variável. E também possível ter mais de um "wire" em cada lado de uma célula, permitindo superposição de fios ou células mais complexas.

Os operadores a serem aplicados às células podem ser: LEFT, RIGHT, ABOVE, BELOW, ROTATED0, ROTATED90, ROTATED180, ROTATED270, FLIPPED0, FLIPPED90, FLIPPED45, FLIPPED135.

A subrotina extcell especifica um arquivo contendo uma célula externa. A célula externa pode estar em forma intermediária, em cujo caso a chamamos "flexível", ou pode estar em CIF, quando então a chamamos "rígida". Um tipo especial de células externas são "pads", que são células rígidas. Existe uma biblioteca de "pads".

As subrotinas begincell e endcell são usadas para delimitar uma célula composta. O parâmetro de begincell é uma sequência de caracteres contendo o nome da célula; o nome pode conter somente brancos, caso não se queira atribuir nome à célula. O nome é usado para localizar erros.

O parâmetro de endcell é uma sequência de caracteres contendo os nomes dos fios que constituem a interface da célula. Caso esses nomes sejam especificados, eles aparecerão no código CIF, e são úteis para simulação.

A figura 6 mostra um programa que descreve uma árvore binária de células "nand" e o layout correspondente. Assume-se que a célula flexível "nand", como mostrada na figura 1, foi gerada anteriormente.

```
program binarytree(output);
const
#include "/va/allende/usr/const.h"
type
#include "/va/allende/usr/type.h"
#include "/va/allende/usr/proc.h"
procedure root;

begin
  extcell('nand.pif');
end;

procedure btree(n: integer);
begin
  begincell('btree');
    if n = 1 then root
    else begin
      root;
      place(ABOVE);
      begincell('  ');
        btree(n-1);
        place(LEFT);
        btree(n-1);
      endcell('  ');
    end;
  endcell('  ');
end;

begin
  btree(3);
end.
```

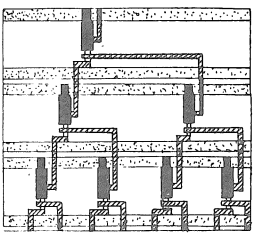


Fig. 6 - Programa para uma árvore binária

2.3. A forma intermediária PIF

PIF consiste em uma forma compacta de representar a estrutura do layout, como na figura 3. Os objetos no layout são células, e os operadores especificam posição ou orientação. Nossa representação do layout é exatamente como uma expressão aritmética; operandos são

células, operadores binários especificam posição relativa (esquerda, direita, acima, abaixo) e operadores unários especificam orientação (rotação ou espelhamento). A precedência de operadores é a usual, e parênteses podem ser usados para mudar precedência.

Um layout em PIF é uma combinação estruturada de células, ao passo que um layout em CIF é uma combinação de retângulos e outros elementos em qualquer ordem. O resultado da interpretação de um programa PIF é um conjunto de equações lineares, o layout em forma simbólica (somente coordenadas relativas) e o circuito (a nível de chaves) para simulação.

A figura 7 mostra um programa PIF, gerado pelo programa ALLENDE da figura 5. O código "C [. . d2 m3]" representa uma célula que é o contato de dois fios: um em "diffusion" com largura 2 lambda vindo da direita e outro em metal com largura 3 lambda vindo do lado inferior; os dois "."s indicam nenhum fio nos lados esquerdo e superior. "A" significa "above", "+" significa "crossing", os parênteses delimitam uma célula, e "$example" especifica o nome "example" para a célula.

```
$example
(
C [ . . d2 m3 ]
A
+ [ p2 m3 p2 m3 ]
)
```

Fig. 7 - Um programa PIF

Em PIF, a construção do layout é como avaliação de expressão. Se usarmos uma gramática para descrever a linguagem PIF, a construção do layout pode ser feita em paralelo ao "parsing", de uma forma "bottom up". Em realidade, isto é similar à forma de concepção e implementação do sistema de formatação de textos matemáticos EQN[2]. Em EQN equações são vistas como um conjunto de "caixas" combinadas de diversas formas.

2.4. Layouts através de equações lineares

Como mostrado na figura 4, em nosso sistema de layout existem diferentes representações para o layout: a representação do usuário (em ALLENDE), a forma intermediária (em PIF), a forma simbólica (em termos de coordenadas relativas) e a forma absoluta (em CIF).

Nossa representação simbólica do layout é em termos das coordenadas relativas dos elementos do layout; a relação entre estas coordenadas é expressa por um conjunto de equações lineares. As variáveis nessas equações são as coordenadas X e Y dos objetos no layout. As equações descrevem a interação entre os objetos, e podem vir das regras de desenho, conectividade e hierarquia na descrição do layout. Resolvendo o sistema de equações e substituindo os valores obtidos para as coordenadas no layout simbólico obtemos o layout absoluto.

O conjunto de equações é resolvido de forma a minimizar a área total. Devido ao grande número de elementos no layout, a forma das equações deve ser tão simples quanto possível, para reduzir a complexidade do algoritmo de solução. Assume-se que equações em X e Y são desacopladas: nenhuma equação envolve ambas as coordenadas X e Y, e equações envolvendo X e Y são independentes. Não se permite relacionar equações através do operador or, por exemplo. Com o desacoplamento das equações em X e Y o problema de compactação é equivalente a resolver dois sistemas de equações independentes.

O layout completo é representado usando equações da forma:

$$x_i = x_j$$
$$x_i - x_j > d \qquad (d > \emptyset, \text{ inteiro})$$
$$x_i - x_j = e \qquad (e > \emptyset, \text{ inteiro})$$

Temos um algoritmo eficiente para resolver tais equações, descrito em [4]. O algoritmo é baseado na classificação topológica.

Cada equação $x_i - x_j = e$ corresponde a uma célula rígida. O usuário pode controlar o número de equações construindo o layout em diversas etapas: obtendo células rígidas e usando-as no próximo nível da hierarquia de células. Se não existem equações da forma $x_i - x_j = e$ no conjunto de equações geradas, sempre existe uma solução para as equações, visto que as equações são geradas de forma a não criar "ciclos". A única situação em que não há solução para o conjunto de equações ocorre quando uma célula rígida não cabe no contexto em que ela é instanciada. Por exemplo, alguma condição pode forçar uma separação maior entre dois fios na interface de uma célula rígida.

O sistema de layout ALLENDE compreende atualmente os seguintes programas:

- ALLENDE
  Ele toma um programa em Pascal ou C e gera o layout (célula rigida), o circuito a nível de chaves, ou uma célula flexível para ser usada posteriormente.

- SIMULATE
  Este programa é um simulador a nível de chaves, baseado em [6].

- CIF2CELL
  A idéia de CIF2CELL é tornar possivel o uso de código CIF gerado por outras ferramentas. Ele basicamente determina a interface da célula. O código CIF é então usado como uma célula rigida.

- CIF2CIRCUIT
  Quando células rigidas são usadas, não é possivel extrair o circuito da descrição em "alto nivel" do layout. Nesse caso, o circuito é extraido do layout em CIF.

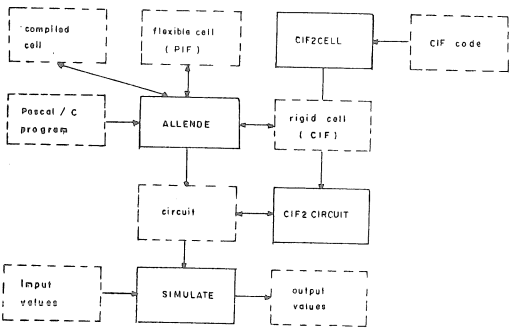


Fig. 8 - O sistema ALLENDE

Este sistema de layout funciona para a tecnologia nMOS, e extensões para outras tecnologias estão sendo estudadas. Os programas que compõem o sistema foram escritos em C e Pascal. O sistema roda sob o sistema operacional UNIX, em um computador VAX. Para fins de utilização deste sistema de layout no Brasil, está sendo providenciado o transporte desse sistema para o sistema operacional MVS, no mesmo VAX, e possivelmente para microcomputadores.

3. Vantagens de nosso método

- **concepção hierárquica**
  A linguagem ALLENDE força uma concepção hierárquica, e a representação textual do layout reflete esta hierarquia.

- **poder expressivo**
  Todo o poderio de uma linguagem "procedural" está disponivel ao projetista.

- **parametrização**
  Diferentes layouts podem ser obtidos simplesmente mudando um parâmetro no programa.

- **documentação**
  O programa é uma documentação do projeto.

- **ferramenta aberta**
  Editores gráficos tendem a ser ferramentas fechadas no sentido de que é difícil automatizar o processo de layout além do que o projeto original do sistema permite. Linguagens "procedurais" são melhores neste aspecto. A entrada para um compilador é texto que pode ser gerado por seres humanos ou por um programa, enquanto um editor gráfico tem uma natureza interativa, sendo projetado basicamente para aceitar comandos gerados por seres humanos.

- **desnecessários equipamentos caros**
  Não há necessidade de sofisticados terminais gráficos.

- **células flexíveis**
  Posições e tamanhos absolutos das células são determinados somente depois de especificadas as posições de todas as células, eliminando problemas de compatibilidade de interface entre células.

- obediência a regras de desenho
  As regras de desenho estão implícitas no processo de geração de equações, garantindo que o layout obtido não contém violações a essas regras.

4. Conclusões

O sistema de layout ALLENDE tem sido usado por um grande número de pessoas, em universidades e centros de pesquisa dos Estados Unidos. Diversos circuitos foram projetados e fabricados com sucesso. ALLENDE foi usado também em experimentos que requerem a produção de diversos layouts diferentes para um mesmo tipo de circuito [1]. Algumas ferramentas de layout, como um gerador de PLA (programmable logic array) a partir de equações booleanas e um roteador de "pads", foram escritas em ALLENDE com esforço mínimo.

Um aspecto importante de um sistema, visto somente quando o sistema é usado, é a deteção e localização de erros. Em ALLENDE os layouts produzidos são corretos por definição, em termos de conectividade e regras de desenho. Em caso de erros na especificação do layout pelo usuário, o sistema aponta as células envolvidas no erro.

Quanto a compactação, os layouts produzidos pelo sistema são relativamente densos. É difícil fazer uma comparação de densidade para layouts produzidos por ALLENDE e layouts produzidos manualmente, porque isto depende da regularidade do layout e da engenhosidade do projetista. Baseado nos layouts já produzidos, encontramos que para estruturas regulares a densidade é praticamente a mesma, enquanto que para estruturas irregulares gastamos aproximadamente 20% mais área do que layouts compactados manualmente.

A representação estruturada do layout e o uso de uma linguagem intermediária (PIF) levaram a um sistema bastante eficiente, em termos de gasto de memória e tempo de execução, e uma implementação simples.

Em um programa ALLENDE a estrutura do circuito e a estrutura do layout se sobrepõem, isto é, o usuário descreve ao mesmo tempo a estrutura do circuito e a estrutura do layout. A estrutura do circuito vai até o nível de transistores e contatos. Poderíamos ter uma linguagem permitindo a especificação do circuito em um nível mais alto (a nível de porta ou a nível funcional, por exemplo). Desta

especificação o layout em PIF seria gerado. Generalizando, PIF (ou ALLENDE) poderia ser a linguagem objeto para uma ferramenta de projeto de circuitos integrados, até mesmo um compilador de silício.

Além de ser uma ferramenta poderosa, ALLENDE está associada a uma metodologia estruturada de projeto de circuitos integrados. Uma ferramenta que força hierarquia e o uso de estruturas regulares vai melhorar a forma como são projetados circuitos integrados. Este é um passo na direção do domínio da complexidade do projeto de circuitos integrados.

5. Referências

[1] K. Iwano and K. Steiglitz
"Some Experiments in VLSI Leaf-cell Optimization."
1984 IEEE Workshop on VLSI Signal Processing, Los Angeles, CA, Nov. 1984.

[2] B. Kernighan and L. Cherry
"A System for Typesetting Mathematics"
Communications of the ACM, Vol. 18, No. 3, March 1975.

[3] R.J.Lipton, J.Valdes,G.Vijayan,S.C.North, and R.Sedgewick
"VLSI Layout as Programming"
ACM Transactions on Programming Languages and Systems, Vol. 5, No. 3, July 1983.

[4] J. M. Mata
"Solving Systems of Linear Equalities and Inequalities Efficiently"
15th Southeastern Conference on Combinatorics, Graph Theory and Computing, Baton Rouge, LA, March 1984.

[5] J. M. Mata
"A Methodology for VLSI Design and a Constraint-Based Layout Language"
Ph.D. Thesis, Princeton University, Oct. 1984.

[6] V. Ramachandran
"An Improved Switch-Level Simulator for MOS Circuits"
20th Design Automation Conference, Miami Beach, FL, June 1983.

[7] J. Valdes and R. L. Kalin
"ALI2 Documentation and Implementation Guide: Language Overview"
VLSI Memo #8, Princeton University, Feb. 1983.

O ROTEADOR DE CANAL DO CONJUNTO DE FERRAMENTAS DE PROJETO DO NCE/UFRJ.

M.F. MARTINS* e E.A. SCHMITZ**

SUMÁRIO

Este artigo apresenta os algoritmos, a implementação e os resultados do programa roteador de canal que faz parte do conjunto de ferramentas do projeto de CI's do NCE/UFRJ.

ABSTRACT

This paper describes the algorithms, implementation and results of the channel router program, which is included in the IC design tool set of the NCE/UFRJ.

* Engenheiro eletrônico (UFRJ, 1972), M.Sc. COPPE - UFRJ, 1975; Projeto Circuitos Digitais;

** Engenheiro eletrônico (UFRGS, 1971), M.Sc. COPPE - UFRJ, 1973; Ph.D. Imperial College 1980; Projeto de Circuitos Integrados.

Endereço de ambos os autores: Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, CCMN, Cidade Universitária - Ilha do Fundão, Caixa Postal 2324 , CEP 20001, Rio de Janeiro, telefone (021) 290-3212, ramal 290 e 285.

O conjunto de ferramentas de projeto do NCE/UFRJ contem programas para executar diversas tarefas encontradas no decorrer do projeto de um circuito integrado LSI.

Como ferramentas típicas temos: simulador funcional, editor gráfico, gerador de módulos, verificador de regras de projeto, extrator, simulador e programas de plotagem. Estas ferramentas permitem definir, projetar e verificar um módulo ou um conjunto de módulos. Para completar este conjunto de ferramentas sentimos a necessidade de um programa de roteamento automático entre módulos.

Os atributos requeridos deste programa em ordem de importância são:
1. Integração com o conjunto existente de ferramentas. Isto implica que o programa seja compatível com o editor gráfico EDCI e que dê como resultado um arquivo no formato CIF;
2. Execute rapidamente de forma a poder oferecer ao projetista a opção de tentar vários arranjos de planta-baixa e escolher o que oferece a melhor forma de conexão.

Acreditamos que ambos os objetivos foram plenamente alcançados, como tentaremos mostrar no restante deste artigo.

O PROBLEMA DO ROTEAMENTO DE CANAL

Consideremos um canal retangular com 2 linhas de terminais uma no topo e a outra na base. A cada terminal é assinalado um número inteiro de Ø a N sendo o Ø usado para representar ausência de conexão. Terminais com o mesmo número i (1 < i < N) são ligados entre si por meio da rede i.

Duas camadas são disponíveis para o roteamento: uma para as trilhas horizontais e a outra para as verticais estando dessa forma estas isoladas daquelas. A conexão entre uma trilha horizontal e uma vertical é feito então por meio de um furo. Cada rede possui no máximo uma trilha horizontal ligada aos terminais superiores e inferiores por meio de trilhas verticais. O problema é expresso por meio de uma lista de redes como mostra a figura 1 e uma solução aparece na figura 2.

0	1	4	5	1	6	7	0	4	9	10	10
2	3	5	3	5	2	6	8	9	8	7	9

Fig. 1

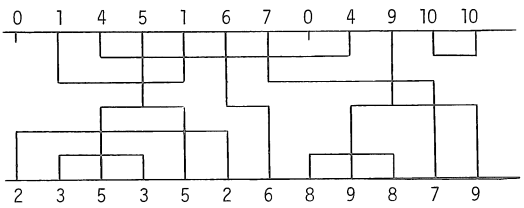
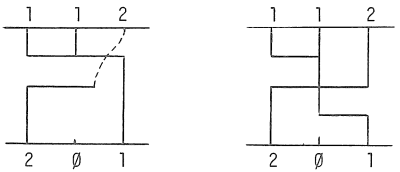


Fig. 2

Um problema que pode surgir durante o roteamento é denominado "conflito cíclico" exemplificado abaixo.

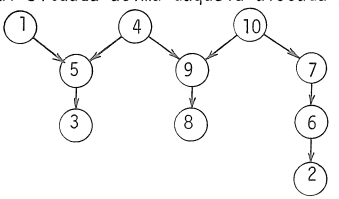


Neste caso uma solução seria alterar a ordem de sinais, o que é relativamente simples em se tratando de PLA's ou alocando mais de uma trilha horizontal para um ou mais sinais. O programa que desenvolvemos possui uma rotina para deteção de ciclos imprimindo ao primeiro ciclo detetato a sequência de redes que o compõe facilitando assim uma alteração na ordem dos sinais, abortando em seguida.

O objetivo do roteamento é minimizar o número de trilhas horizontais de modo que a largura e consequentemente a área do canal seja mínima.

RESTRIÇÕES VERTICAL E HORIZONTAL

Supondo que há somente um segmento horizontal para cada rede é claro que o segmento horizontal de uma rede conectada ao terminal superior de uma coluna qualquer deve estar situado acima do segmento correspondente a rede conectada ao terminal inferior daquela coluna. Esta relação pode ser representada por meio de um grafo orientado onde cada nó representa uma rede e cada seta dirigida de a para b significa que a rede a deve utilizar uma trilha horizontal situada acima daquela alocada para b.



Em geral quando há um caminho de cima para baixo de i para j dizemos que i é ancestral de j e j é descendente de i.

REPRESENTAÇÃO POR ZONAS

O segmento horizontal de uma rede é delimitado por suas conexões mais a esquerda e mais a direita. Sendo S(i) o conjunto de redes cujos segmentos interceptam a coluna i, desde que os segmentos de redes distintas não devem se superpor, para cada rede em S(i) devemos alocar uma trilha horizontal. Esta condição deve ser satisfeita para cada coluna mas, como é fácil ver, basta considerar apenas os conjuntos S(i) que não são sub-conjuntos um do outro. Para cada um deste conjunto corresponde um certo grupo de colunas ao

lingo do canal e que chamamos de zona indicada pela tabela abaixo. A direita aparece uma representação mais completa onde podemos visualizar que redes começam e/ou terminam em cada zona.

Coluna	S(i)	Zona
1	2	
2	1 2 3	
3	1 2 3 4 5	1
4	1 2 3 4 5	
5	1 2 4 5	
6	2 4 6	2
7	4 6 7	3
8	4 7 8	
9	4 7 8 9	4
10	7 8 9	
11	7 9 10	5
12	9 10	

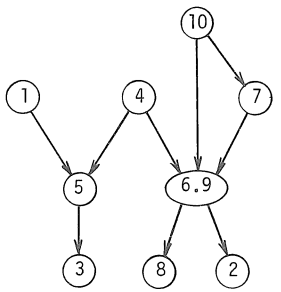
zona

1	2	3	4	5
1		7		
2			8	
3			9	
4				10
5	6			

FUSÃO DE REDES

Antes de descrever o algoritmo definiremos a seguinte operação:
Sejam 2 redes i e j tais que:
1. Não há superposição horizontal na representação em zonas;
2. Não há um caminho orientado entre os nós i e j no grafo.

A fusão da rede j na rede i (dizemos que j foi "absorvido") consiste então em:
1. Modificar o grupo de restrição vertical "deslocando" o nó j até superpô-lo com o nó i. Consequentemente acrescentamos ao nó i os descendentes de j. Além disso todos os ancestrais de j passam a ser ancestrais de i;
2. Na representação em zonas a rede resultante passa a ocupar 2 zonas consecutivas (as zonas de i e de j).

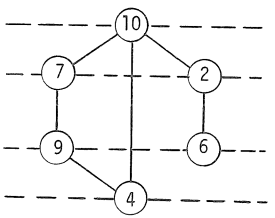


zona

1	2	3	4	5
1		7		
2			8	
3				
4				10
5	6.9			

O algoritmo a seguir funde redes sistematicamente de acordo com a representação em zonas.

O grupamento dos nós por geração não é tão trivial como pode parecer a primeira vista. O nó 11 e o nó 4, embora sejam ambos filhos do nó 3, não pertencem a mesma geração uma vez que existe um caminho do nó 11 até o nó 4. Um outro exemplo, ainda mais significativo, aparece abaixo. Embora haja um caminho direto do nó 10 (no topo) até o nó 4 este encontra-se hierarquicamente abaixo do nó 6 onde não há tal caminho. O critério para definição das gerações consiste então em considerar sempre os caminhos mais longos.



No algoritmo para alocação de trilha que aparece a seguir foi empregado o termo descendente para indicar que existe algum caminho entre 2 redes e o termo filho no sentido restrito em que a distância é igual a 1.

```
t = 1;
P = [redes do topo do grafo];
Enquanto P ≠ [] faça
   Início
      Q =[ ];
      Para cada rede r em P faça
         Início
            Atribuir trilha t para todas as redes fundidas com r;
            t = t + 1;
            Q = Q + [filhos de r]
         Fim.
      Para cada rede r em P faça
      P = P - [descendentes de r]
   Fim.
```

OTIMIZAÇÃO DAS ROTAS

O programa de roteamento tinha como objetivo inicial apenas o de minimizar o número de trilhas horizontais e com isso a área do canal. Observando, no entanto, diversos resultados verificou-se que embora esta condição fosse satisfeita ocorria que algumas redes se apresentavam desnecessariamente longas como mostra o exemplo a seguir.

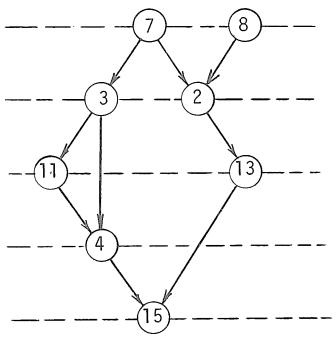
```
L = [ ];
para z de zi até zf faça
  Início
     L = L + [redes que terminam na zona z],
     R = [redes que começam na zona z + 1];
     Fundir redes de L com redes de R de modo a minimizar o aumento do mais longo cami
     nho no grafo;
     L = L - [redes fundidas no passo anterior]
  Fim.
```

Antes de prosseguir devemos observar pelas definições de L e R que nunca haverá restrição horizontal entre uma rede qualquer de L e uma qualquer de R. O passo chave do algoritmo é o que realiza a função entre as redes de L e R. Se há um caminho $n_1 - n_2 - \ldots n_k$ no grafo de restrição vertical então 2 quaisquer dessas redes não podem ocupar a mesma trilha horizontal, ou seja, se no mais longo caminho o número de nós é k pelo menos k trilhas serão necessárias para fazer as interconexões. A fusão de redes deve portanto ser feita de modo que o mais longo caminho após a fusão seja tão pequeno quanto possível.

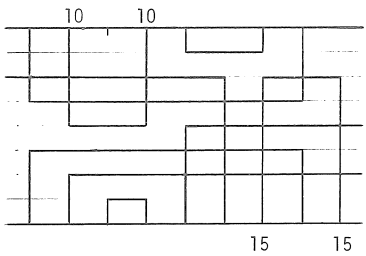
Para minimizar o mais longo caminho usamos uma série de considerações heurísticas descritas no artigo de Takeshi Yoshimura e Ernest S. Khu: "Efficiente Algorithms for Channel Routing". (IEEE TRANS on CAD, V. 1, No. 1, JAN 82, pp. 25-35).

ALOCAÇÃO DE TRILHAS

Uma vez chegando ao grafo na sua forma mais simples, ou seja, com o menor número possível de nós resta então fazer a alocação de trilhas. Para cada nó - que na verdade corresponde a um grupo de redes que se fundiram - devemos alocar uma trilha. Essa alocação deve levar em conta a hieraquia entre os nós estabelecida pelo grafo. Devemos portanto começar pelo topo do grafo identificando a partir daí cada uma das gerações até chegar à base como indicado abaixo.

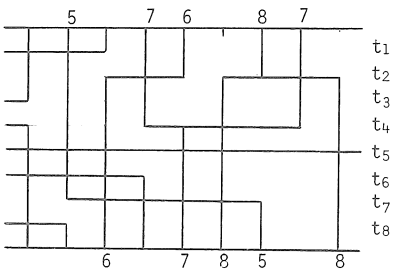


GERAÇÃO	NÓS	TRILHAS
1	7,8	1,2
2	2,3	3,4
3	11,13	5,6
4	4	7
5	15	8



A rede 10 poderia ocupar a trilha 1 em vez da 3 e a rede 15 ocupar a trilha 7 em vez da 2. Visualmente é simples fazer o deslocamento para cima ou para baixo mas como programar este tipo de tarefa? Antes de mais nada o sentido do deslocamento depende das conexões da rede ao longo do canal; uma rede deve ser deslocada para cima ou para baixo conforme o número de conexões para cima ou para baixo seja maior respectivamente. Toda rede com igual número de conexões não necessitaria se deslocar. Foi criado então um vetor (CONEXÕES: ARRAY [0 ... MAX REDES] OF INTEGER) inicializado com tudo zero. A rotina LE DADOS ao ler o arquivo de entrada "fios.dat" vai incrementando ou decrementando cada elemento do vetor e no final teremos então, associado a cada rede, um inteiro positivo, zero ou negativo indicando qual o deslocamento a ser feito.

Uma vez identificadas as redes a serem deslocadas e qual o sentido de deslocamento o passo seguinte consiste em determinar, para cada uma delas quais as redes que limitam seus deslocamentos. Este tipo de problema aparece ilustrado abaixo.



A rede 7 deve ser deslocada para cima o máximo possível. Acima dela estão livres as trilhas 1 e 3. A trilha 1, no entanto, não pode ser aproveitada por causa da rede 6 que funciona como um "bastante" impedindo que a rede 7 atinja aquela trilha. Da mesma forma a rede 8 não poderia ocupar a trilha mais baixa por causa da rede 5. Deveria ocupar então a trilha 6.

Este segundo passo sugere o uso de um vetor de vetores: a cada rede é associado um conjunto de redes "batentes". Na prática mostrou-se mais conveniente o uso de um vetor de records tendo por componentes um vetor e uma escala indicando o número de redes "batentes" encontradas.

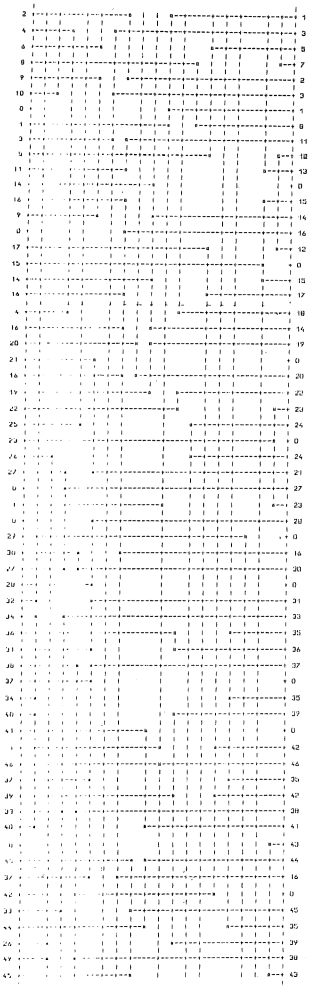
Quando uma rede qualquer tem maior número de conexões para cima ela deve ser deslocada para cima e neste caso devem ser listadas todas as redes ao longo do canal que apareçam acima dela. Se o número maior de conexões for para baixo devem ser listadas aquelas que pareçam abaixo. Em um trecho da procedure CRIAGRAFO todo o canal é percorrido e para cada rede encontrada (in ou sup) um dos testes é feito: conexões [inf] > 0 ou conexões [sup] < 0. No primeiro, se verdadeiro, a rede batente será sup e no segundo será inf.

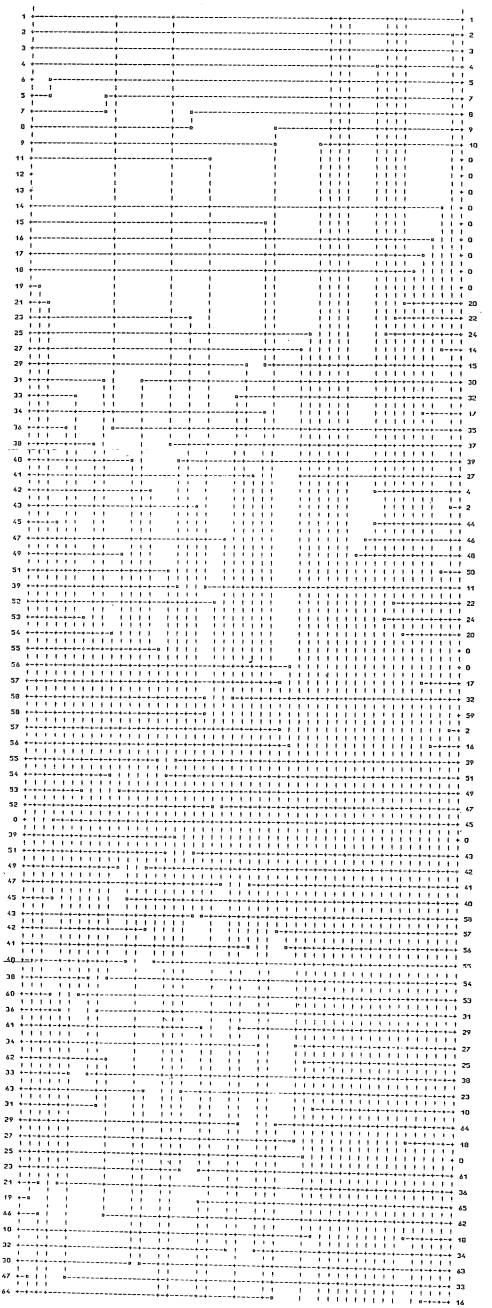
O próximo passo consiste em reunir em um vetor todas as redes a serem deslocadas no mesmo sentido (para cima ou para baixo). Estas redes serão então deslocadas uma a uma a partir do primeiro. Antes disso porém fazemos uma primeira ordenação das redes em ordem crescente de largura e a seguir uma ordenação em ordem decrescente de saldo de conexões. Teremos, após estas 2 ordenações, situadas em primeiro lugar na lista as redes com maior saldo de conexões e portanto com maior redução no comprimento devido ao deslocamento. Se 2 ou mais redes tem mesmo saldo serão deslocadas primeiro aquelas com menos largura e portando maior probabilidade de encontrar trilhas disponíveis.

IMPLEMENTAÇÃO

O programa foi escrito em Pascal, tendo aproximadamente 1300 linhas. Apresenta a saída em dois formatos: a primeira em impressora e a segunda na forma de um arquivo CIF contendo a descrição do lay-out na tecnologia NMOS.

As figuras seguintes mostram exemplos de dois canais gerados pelo programa, com tempos de execução iguais a 1,5 minutos para 17 trilhas e 2,5 minutos para 45 trilhas (num microcomputador com processador 68000 de 8MHZ).





EXTRAI - EXTRATOR DE CIRCUITOS INTEGRADOS NMOS.

A.A.S. TELES* e L.F.P. SOUZA**

SUMÁRIO

O objetivo deste artigo é descrever o algoritmo e a implementação do extrator de circuitos integrados NMOS desenvolvido no NCE/UFRJ, dando ênfase às modificações feitas para a versão 3.0, que permitiram a extração da capacitância dos fios e resistência dos transistores.

ABSTRACT

The intent of this paper is to describe the algorithm and implementation of a NMOS IC extractor developed at NCE/UFRJ. The emphasis is in changes to version 3.0, which allow the extraction of node capacitance and resistance of transistors.

* Mestre em Sistemas Digitais (UFRJ); Inteligência Artificial, Compiladores, Cad para VLSI; Núcleo de Computação Eletrônica da UFRJ, Cidade Universitária, CCMN, Rio de Janeiro, Caixa Postal 2324, CEP 20001, telefone (021) 290-3212, ramal 293;

** Estudante de Informática (UFRJ); Software Básico, Sistemas Operacionais, Compiladores, Cad para VLSI; Núcleo de Computação Eletrônica da UFRJ, Cidade Universitária, CCMN, Rio de Janeiro, Caixa Postal 2324, CEP 20001, telefone (021) 290-3212, ramal 293.

INTRODUÇÃO

No Sistema de Apoio a Projetos em Circuitos Integrados NMOS atualmente implementado no NCE/UFRJ, adotou-se a metodologia Mead & Conway em que o circuito é projetado em função de uma unidade padrão chamada Lambda, que varia de fabricante para fabricante. A primeira fase do projeto consiste na definição do funcionamento do chip numa linguagem lógica e a sua simulação funcional (SIMF). A partir do circuito validado, as PLA's são geradas automaticamente (MKPLA), e podem ser aproveitados trechos padrões (biblioteca de célula), entretanto grande parte dos blocos funcionais e todas as suas conexões precisam ser feitas através de editores gráficos (EDMOS e EDCI). Terminada a edição o projetista tem um banco de dados que armazena hierarquicamente a geometria do circuito, baseado na linguagem Cif (formato intermediário da Caltech), e poderia gerar uma fita com a descrição em Cif e enviá-lo ao fabricante, entretanto como a edição é um processo manual faz-se necessário algumas ferramentas de validação, como verificador de regras estáticas elétricas e lógicas (ESTÁTICO), simulador lógico (SIML), simulador elétrico (SPICE) e avaliador de desempelho (CRÍTICO).

Como as ferramentas de verificação lógica e elétrica não manipulam eficientemente uma descrição geométrica, torna-se necessário a construção um mecanismo que a partir do lay-out extraia uma descrição a nível de fios e transistores. Tanto o extrator (EXTRAI) como o verificador de regras geométricas (DRC) precisam da mesma imagem não hierárquica do lay-out, então foi criada uma ferramenta intermediária (GMAP), que do banco de dados gera a imagem rastreada do circuito e um arquivo com as coordenadas dos pontos nomeados pelo projetista. Estes dois arquivos constituem a entrada do extrator.

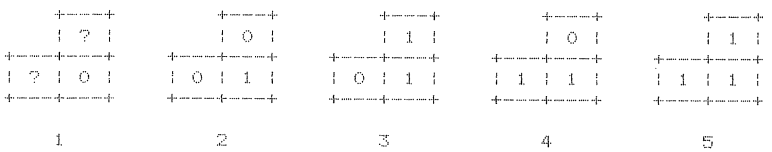
DESCRIÇÃO

Projetar um circuito integrado é basicamente determinar quais as máscaras geométricas a serem usadas na construção de cada camada. As camadas na tecnologia NMOS são, difusão, poli-silício, metal, corte e implante. As camadas difusão, poli-silício e metal são condutoras, a camada de corte faz contato entre uma camada de metal e outra de poli-silício ou difusão. Quando um fio de poli-silício cruza um de difusão, forma-se um transistor onde o poli-silício é o gate e cada pedaço da difusão um dos drenos.

Os editores gráficos criam uma descrição do circuito em CIF a partir dos lay-outs projetados. Esta descrição pode ser representada por um mapa de pixel, onde cada byte é um quadrado de lambda por lambda do lay-out e seus bits indicam a existência ou não de cada camada neste ponto.

O mapa de pixel descrito acima é a principal entrada do extrator, e nele é aplicado o algoritmo de ilhas e lagos descrito em BACKER {1}. O algoritmo procura resolver o seguinte problema, dada uma matriz de zeros e uns, onde os uns indicam terra e os zeros água , atribuir um código a cada região contínua de terra e determinar sua área. A solução é um autômato bidimensional que varre o circuito da esquerda para direita e de cima para bai

xo atribuindo um código a cada posição, este código é função do código da posição imediatamente acima e do código da imediatamente a esquerda. O código zero indica que a região é água. A função é descrita abaixo para cada configuração.



1 - Se a posição atual na matriz for zero, o código correspondente é zero por definição;
2 - Se a posição atual for um e a superior e esquerda forem zeros, uma nova região foi encontrada, logo é criado um novo código e atribuído a este ponto;
3 - Se a posição atual e superior forem um, e a esquerda zero, significa que se está sobre uma aresta vertical de uma região, logo o código da posição atual será o mesmo da posição superior;
4 - Se a posição atual e a esquerda forem um, e a superior for zero, significa que se está sobre uma aresta horizontal de uma região, logo o código da posição atual será o mesmo da posição esquerda;
5 - Se a posição atual, a superior e a esquerda forem um, temos dois casos:
   a) Os códigos das regiões superior e esquerda são os mesmos, logo se está dentro de uma região contínua, o código do ponto será o código desta região, ou seja, o código das posições superior e esquerda;
   b) Os códigos são diferentes, então se está sobre a união de duas partes da mesma região em formato de um L invertido, logo deve ser feita uma equivalência entre estas duas regiões, e o código equivalente será atribuído a posição atual.

Como foi mencionado deve ser montada uma relação de equivalência, que terá suas características discuticas abaixo.

O algoritmo descrito acima extrai apenas fios. Entretanto o mesmo autômato pode ser usado para identificar regiões periféricas aos transistores, desta forma numerando-os e determinando suas portas e drenos, além de fornecer elementos para determinar suas dimensões. Algumas destas situações são indicadas a seguir, onde D indica difusão, P poli-silício e T difusão + poli-silício.

A capacitância dos fios e transistores, é calculada como o somatório da capacitância em cada pixel, sendo esta função da configuração das camadas neste pixel. Esta capacitância é devida a diferença de potencial entre as camadas condutoras e o substrato, que é a base da fundição: e entre as camadas de metal e poli-silício ou metal e difusão, como explicado abaixo.

Como as camadas difusão, poli-silício e metal ficam isoladas do substrato pode existir uma diferença de potencial entre cada uma destas camadas e o substrato, o que implica numa capacitância proporcional a área. Como a camada de difusão fica enterrada no substrato também deve ser somada uma capacitância proporcional ao perímetro em difusão de cada nó, esta capacitância é também proporcional a profundidade da camada de difusão. Quanto a capacitância que pode aparecer entre o metal e o poli-silício, ou entre o metal e a difusão, deve ser tomado o seguinte cuidado, se as regiões forem as mesmas, isto é, existir um contato entre elas como da camada de metal para o poli-silício no esquema, a capacitância será nula, mas se as regiões forem diferentes existirá capacitância, a implementação deste cálculo deve prever que o contato geralmente aparece depois que algumas posições já foram analisadas.

IMPLEMENTAÇÃO

Na implementação do algoritmo de ilhas e lagos só precisamos ter na memória uma linha do mapa de pixel's de cada vez, indicando a existência ou não de cada camada em cada ponto

desta linha. Paralelo a esta linha teremos uma matriz que representa uma função que dado um ponto e uma camada fornece qual o código da camada naquele ponto. O autômato em que se baseia o algoritmo necessita do código da posição superior e da esquerda ao ponto que se está analisando, entretanto a matriz que armazena os códigos não precisa possuir os códigos destas duas linhas, basta que se considere os códigos a direita da posição sendo analisada como referentes a linha superior, e que coloque os novos códigos na matriz a medida que se anda para direita. Desta forma os códigos a esquerda da posição sendo analisada se referirão a linha atual. Na realidade como o mapa é percorrido no mesmo sentido que um texto só necessitamos ter a cada momento na memória um pixel, e a matriz que representa os códigos, como tem seu acesso bastante localizado, se presta otimamente para mapeamento.

Para podermos representar os códigos dos transistores e tratar as configurações das camadas de forma melhor, criamos uma pseudo-camada que chamamos de TRANSISTOR e modificamos os bits de cada ponto do mapa em análise. Esta modificação faz com que o mapa represente mais fielmente as disposições físicas reais das camadas. As modificações são duas: liga-se o bit que indica a presença de transistor nos pontos onde ocorrem difusão e poli-silício, e apaga-se o bit de difusão nos pontos onde ocorre poli-silício, o que se aproxima mais da construção das portas (gates) como indicado abaixo.

A relação de equivalência necessária para o algoritmo de ilhas e lagos é uma relação reflexiva, simétrica e transitiva. Toda vez que uma nova região de código N é identificada, obrigamos que N R N, ou seja EQUIV (N) = N. Para cada conjunto de regiões equivalentes teremos uma lista de equivalência construída de tal modo que EQUIV (N) = N. Desta forma, faz-se a equivalência de duas regiões previamente distintas N e M, seguindo as listas de equivalência de N e M até, em cada lista, termos EQUIV (T) = T. Supondo que a lista de equivalência de N leve a uma região Tn e a de M, a uma região Tm, faz-se EQUIV (Tm) = Tn, se Tn < Tm. Caso contrário, faz-se EQUIV (Tn) = Tm. Desta forma garantimos a transitividade da relação, ou seja dada tres regiões A, B e C, se obrigarmos A R B, e mais tarde B R C, automaticamente será montada a relação A R C, independente dos códigos numéricos de A, B e C.

Para a extração dos transistores, utiliza-se o automato do algoritmo de ilhas e lagos, identificando os transistores e suas regiões de porta e drenos. Para isto é montada uma lista onde é depositado um registro para cada pixel de perímetro entre difusão e transis

tor. Desta forma depois de todo o mapa percorrido e esta lista ordenada pelo código do transistor pode se percorrer esta lista e identificar quais as regiões de difusão conectadas a cada transistor. Podem ocorrer então tres casos, se só houver uma região de difusão, então na verdade não se tratava de um transistor real, mas provavelmente um contato entre difusão e poli-silício, que é então desprezado. Se houver duas regiões de difusão então é um transistor normal, se houver mais de duas regiões, então é um transistor, que embora teoricamente possível, deve ser indicado como inválido.

Outras informações também são extraídas dos transistores, a existência de implante, a existência de corte junto ao transistor, o perímetro entre poli-silício e transistor, a área do transistor e a direção de cada fronteira entre difusão e transistor. Com esta direção sabe-se mais tarde se na região de fronteira a difusão estava a esquerda, direita, acima ou abaixo do transistor. A partir destes dados pode se avaliar a capacitância e as dimensões do transistor.

A capacitância de porta do transistor é simplesmente proporcional a área do transistor. Para as dimensões existem dois algoritmos, um para transistores simples, e outro para transistores que apresentem um contato entre porta e dreno, como pull-ups e alguns pull-downs.

Se o transistor não apresenta contato, então considera-se que suas dimensões sejam constantes ao longo de seu comprimento, então a largura é obtida dividindo-se o perímetro em difusão por dois, e o comprimento dividindo-se o perímetro em poli-silício por dois, depois dos comprimentos calculados é feito uma comparação para verificar de o produto da largura pelo comprimento está entre 80% e 120% da área real do transistor, caso esteja fora desta faixa o usuário é advertido.

Nos transistores que apresentam contatos, geralmente estes ocorrem entre a porta e um dos drenos, então o corte fica numa região de fronteira entre o transistor e a difusão, logo a difusão neste ponto é mais larga que no restante do transistor para atender às regras geométricas de construção de contato. O extrator utiliza a informação de direção de cada pixel das fronteiras para separar o perímetro de difusão em que ocorre o corte do perímetro normal, feito isto sabe-se que a largura efetiva será o perímetro da região normal, que é menor que da região em que ocorre o corte, e o comprimento será a metade da diferença entre o perímetro em poli-silício e a extensão do perímetro em poli-silício devida ao alargamento do transistor; esta extensão é calculada pela diferença entre os perímetros em difusão. Caso esta diferença não possa ser calculada (transistor em forma de U) então é assumido o valor 2.

A capacitância é calculada somando em cada região a capacitância de cada pixel, que depende unicamente da configuração das camadas naquele pixel. Entretanto dois cuidados devem ser tomados. O primeiro é quando analizando a capacitância devida a separação do metal e poli-silício ou entre metal e difusão, se as regiões a princípio são diferentes nada impede que ao longo da extração apareça um contato entre estas regiões. Para contor

nar este problema estas capacitâncias não são somadas diretamente aos nos, ao invez disto são colocadas numa lista, que depois de terminada a varredura do mapa é percorrida somando-se aos nos as capacitâncias convenientes.

Outro cuidado que deve ser tomado é quanto a capacitância devida a área e ao perímetro de difusão. Esta capacitância embora pertencente aos nos, e assim tratadas pela maioria das ferramentas, deve ser posta em separado, pois desta forma o simulador elétrico SPICE pode simular mais precisamente a interação entre estas capacitâncias e os transistores.

Os registros do arquivo de saída do extrator podem ser de quatro tipos: dimensionamento, transistor, nos e comentário. Os registros de dimensionamento informam quantos transistores compõem o circuito, antes que o primeiro registro de transistor apareça, idem para os nós. Isto é muito útil nos programas que alocam memória dinamicamente.

Os registros de transistor fornecem os seguintes dados: para cada transistor, tipo (Depletion ou Enhancement), código do transistor, código da região de gate, códigos dos drenos, comprimento, largura, caracter de consistência das dimensões (advertência), coordenadas horizontal e vertical e capacitância de gate (aF).

Os registros de nos possuem, tipo do no (Input ou Normal), nome do no, código, coordenada horizontal e vertical, capacitância total, inclusive a devida a difusão (aF), área em difusão e perímetro em difusão.

O nosso formato de saída não é compatível com o de entrada do SPICE, por isto foi criada um filtro que além de modificar o formato, distribui a capacitância devida a difusão pelos transistores, ponderadamente de acordo com a largura de cada transistor.

CONCLUSÃO

A principal preocupação na implementação do extrator foi o tempo de processamento. Desta forma obrigamos a todos os arquivos sequenciais a serem buferizados e todos os de acesso direto a serem mapeados na memória principal, onde também foram colocadas inteiramente as estruturas de tamanho médio frequentemente acessadas. Daí vem as duas limitações do programa, a largura do circuito deve ser menor ou igual a 3000 lambdas e o número de transistores, verdadeiros e falsos como os contados de emenda, não deve ultrapassar a 20000. A redução do tempo de processamento foi de 5 horas e 30 minutos, da versão anterior, para 1 hora e 20 minutos de CPU do PDP 11/70 da DEC, para um circuito de 1900 lambdas por 1900 lambdas.

Outro objetivo que era passar a extrair a capacitância dos nos e transistores, foi também conseguido. Sendo que a diferença entre a capacitância fornecida pelo extrator e a extraída manualmente por um projetista, é igual a zero, isto é, se o projetista não errar nem fizer algumas simplificações.

Os algoritmos de extração das dimensões dos transistores foram revistos, sendo o resultado atual bastante satisfatório, cobrindo todas as construções usuais.

A resistência entre transistores, devidas aos fios, não é fornecida, devido a limitações de memória e processamento.

Uma nova versão do extrator deverá ser feita ainda este ano, visando o Sistema de Apoio a Projetos CMOS, atualmente fase de definição. Embora a filosofia do novo sistema seja a não intervenção do projetista a nível de transistores e fios, acreditamos que o extrator será necessário tanto na fase de teste do sistema, como na confecção da biblioteca de células básicas a partir da qual o novo sistema montará os circuitos automaticamente.

BIBLIOGRAFIA

BACKER, Clark, "Artwork Analysis Tools for VLSI Circuits", MIT/LCS/TR-239, USA, Massachusets Institute of Technology, 1980;

BRYANT, Randal E., "An Algorithm for MOS Logic Simulation", Lambda, USA, 1(4): 46-53 , 1980;

TELES, Antonio Anibal S., "Verificação de Projetos em Circuitos Integrados", Tese MSc. COPPE - UFRJ, 1983.

FERRAMENTAS CITADAS

01. SIMF - simulador funcional, FORTRAN;
02. MKPLA - gerador automático de PLA's, FORTRAN;
03. EDMOS - editor de células, FORTRAN ou C;
04. EDCI - editor hierárquico de circuitos, C;
05. GMAP - gerador de mapa de pixels, C;
06. DRC - verificador de regras geométricas, C;
07. ESTÁTICO - verificador de regras estáticas lógicas e elétricas, C;
08. SIML - simulador lógico NMOS/CMOS, C;
09. SPICE - simulador elétrico (cedido), FORTRAN e ALGOL;
10. CRÍTICO - avaliador de desempelho (em fase de validação), C.

INTRODUÇÃO AOS CIRCUITOS INTEGRADOS SEMI-DEDICADOS DO TIPO "GATE-ARRAY"

T.F.COSTA *

SUMÁRIO

Neste trabalho procuramos apresentar uma visão geral do que são os circuitos integrados semi-dedicados do tipo "Gate-Array". A organização geral desses circuitos é apresentada, assim como a estrutura de algumas células mais comuns. Finalmente, um roteiro com as principais etapas de projeto é apresentado.

ABSTRACT

In this work we try to present an introduction to the semi-customs IC of the Gate-Array type. The organization of those circuits and the structure of some typical cells are presented. Finally , a description of the design process is also presented.

* Engenheiro Eletrônico (PUC-RJ,77) ; Projeto e Fabricação de Circuitos Integrados.
Laboratório de Microeletrônica da USP - Caixa Postal 8174.
01000 SÃO PAULO - SP - fone: (011) 815.93.22 ramal 310.

Até recentemente a norma em projetos de sistemas digitais era o uso de circuitos integrados (CI) de uso geral ou padrão. O uso de CI especialmente projetados, ou dedicados, era privilégio de alguns grandes fabricantes de sistemas, devido principalmente ao alto custo desses circuitos. No entanto, o grande desenvolvimento em processos de fabricação e recursos computacionais dos últimos anos favoreceu algumas opções de projeto que, tem contribuído para viabilizar o uso de CI dedicados com custos bem menores.

Na figura 1 apresentamos uma classificação dos CI em geral. Inicialmente, temos os CI de uso geral, integralmente projetados e construídos pelos fabricantes de semicondutores para serem comercializados em larga escala. A seguir, temos os CI programáveis, também projetados e comercializados em larga escala pelos fabricantes mas podendo ser personalizados pela programação de curtos ou abertos em determinados nós do circuito. Essa personalização pode ser feita durante a fabricação , na etapa de metalização, ou pelo próprio usuário, através da queima seletiva de fusíveis embutidos no circuito. Finalmente, temos os CI dedicados que, são circuitos projetados para uma função específica e destinados a um único usuário.

Os CI dedicados se dividem em dois tipos: os semi-dedicados e os sob-medida. Os circuitos sob-medida são integralmente projetados para determinada função e normalmente procura-se obter o máximo de desempenho na menor área possível. Envolve portanto, um custo de projeto muito alto em tempo e recursos. Por outro lado, os semi-dedicados constituem opções de projeto que procuram reduzir o alto custo do CI dedicados, limitando a complexidade do projeto.

Os semi-dedicados também se dividem em dois tipos, os pré-fabricados e os pré-caracterizados. Os pré-fabricados são circuitos constituídos de arranjos de componentes pré-definidos e pré-fabricados que podem ser interligados para realizar uma função determinada. Dependendo do tipo de componentes esses arranjos são preferencialmente usado em aplicações lineares ou digitais. Os CI pré-caracterizados são circuitos projetados com um conjunto restrito de funções previamente projetadas e caracterizadas.

Os semi-dedicados pré-fabricados são comumente chamados de arranjos ("Arrays") lineares ou lógicos. Os nomes mais conhecidos são "Linear Arrays" , "Gate-Array" (GA), "Master-Slice" e "Uncommitted Logic Array" (ULA).

As vantagens dos circuitos do tipo GA em relação aos outros tipos de dedicados podem ser resumidas em:

. maior rapidez de projeto e menores custos ;
. maiores chances de sucesso na primeira tentativa ;
. aproveitamento da economia de escala em quase toda a fabricação ;

Como desvantagens temos:

. circuitos de menor desempenho (por usar estruturas pré-fabricadas)
. menor eficiência no aproveitamento de área (circuitos maiores)

TIPOS DE ORGANIZAÇÃO DE GATE-ARRAY:

O projeto de circuitos com Gate-Array é baseado na utilização de um conjunto restrito de componentes agrupados em células dispostas em um arranjo regular e que passaram por todas as etapas de fabricação, exceto as etapas finais de interconexão. A personalização do circuito é obtida finalizando a fabricação com uma ou mais máscaras de metalização, sendo a interconexão de células e componentes determinada pela função a ser realizada. Na figura 2, temos um GA antes e depois da personalização (ref.6).

Existem muitas maneiras diferentes de arranjar as células em um GA (ref.1). As opções mais comuns são mostradas na figura 3. Em um arranjo de blocos, como o da figura 3.a , temos as células dispostas em uma matriz. O espaço entre as células é reservado para a passagem de tiras de interconexão e é comumente chamado de canal de interconexão. No arranjo em fileiras da figura 3.b, as células são colocadas lado a lado, em linha ou coluna, eliminando os canais entre células de uma mesma fileira. Finalmente, no arranjo mais compacto da figura 3.c, as células encaixam uma na outra e os canais são eliminados. Nesse caso, as interligações passam necessariamente através da própria célula. Podemos observar que, nos três casos temos um núcleo de células envolvido por uma periferia e, que, esta também está dividida em células. A função das células de periferia é realizar a interface entre as células internas e o meio externo. Normalmente, são projetadas para trabalhar em níveis de potência bastante superiores ao das células internas, sendo também responsáveis pela compatibilização do circuito interno com as diversas famílias tecnológicas existentes (TTL, ECL, CMOS).

Em geral, dispomos de dois níveis de interconexão para a interligação das células e componentes. O primeiro nível pode ser de difusão ou silício policristalino, mas o segundo é sempre metal (alumínio). Como o metal apresenta condutividade muito superior, procura-se utilizá-lo sempre que possível, deixando o primeiro nível de interconexão apenas para os casos estritamente necessários. Nos GA com arranjos em blocos ou fileiras, as interligações que passam através dos canais são dispostas de forma tal que,

as de um nível sejam ortogonais as do outro, de modo a facilitar o roteamento. Nos GA com arranjo compacto também dispomos de dois níveis, mas as interconexões devem passar através das próprias células e, o roteamento é bem mais complicado que nos casos anteriores. Nos últimos anos aumentou bastante o número de GA com mais de um nível de metal, principalmente entre os bipolares. Um nível adicional de interconexão alivia bastante o problema de roteamento, além de permitir melhorar o desempenho do circuito. No entanto, envolve um processo de fabricação mais complexo e aumenta o número de máscaras de personalização. Enquanto para uma camada de metal uma máscara é suficiente para personalizar um circuito, para dois níveis de metal precisamos de três máscaras, uma para cada nível e uma terceira para as vias de contato entre os dois metais.

Existem GA que não seguem nenhum dos arranjos anteriores mas costumam ter sua área dividida em um certo número de regiões e, cada região tem um conjunto de componentes e arranjo específico. Assim, podemos ter uma região com células dedicadas à implementação de funções especiais, como registradores e memórias. Em alguns casos temos uma ou mais regiões dedicadas à implementação de funções lineares enquanto outras são dedicadas às funções digitais. Outros, adotam um arranjo semelhante ao de fileiras, porém com canais principais, mais largos, separando grupos de fileiras separadas entre si por canais secundários, mais estreitos.

A CÉLULA DO GA

Em geral, as interligações dos componentes que constituem a célula não são totalmente pré-definidas. Faz parte do processo de personalização do circuito a finalização dessas ligações. Assim, o projetista dispõe da possibilidade de, dependendo do caso, determinar a função lógica de cada célula ou pelo menos, algumas de suas características, tais como: número de entradas ou o nível de potência. A estrutura interna da célula é função da família tecnológica e da arquitetura do GA. Existem GA em praticamente todas as famílias tecnológicas e, descrever todas as configurações utilizadas em cada família, ocuparia espaço muitas vezes superior aos limites desse trabalho. Vamos portanto, nos limitar a discorrer brevemente sobre as principais configurações:

- A CÉLULA ECL

O circuito básico de uma célula ECL, figura 4, corresponde a uma porta OR/NOR com saídas complementares (ref-1,7). O funcionamento desse circuito baseia-se em que a corrente I é desviada do transistor Tr para os transistores TA, TB e TC quando o nível da tensão na base desses transistores

passa de um valor inferior a V_{ref} (nível "0") para um valor superior a V_{ref} (nível "1"). O circuito é projetado para trabalhar sempre não-saturado e, em consequência, caracteriza-se por uma alta velocidade e um alto consumo de potência.

Na figura 5, temos alguns exemplos de células ECL práticas (ref.7 a 11). A configuração exata da célula varia bastante, principalmente no que se refere ao número de transistores de entrada. É comum acrescentar seguidores de emissor à célula básica, a fim de compatibilizar os níveis de saída com os necessários na base dos transistores de entrada. Outra variação comum é a implementação da fonte de corrente com um transistor polarizado por uma tensão de referência adicional.

Uma característica do circuito ECL é a utilização de pelo menos uma tensão de referência. Essa tensão é normalmente derivada da tensão de alimentação principal através de geradores de tensão distribuídos entre as células. Uma configuração comum é agrupar duas ou mais células em torno desses geradores. Daí, é comum adotar-se para o GA ECL um arranjo em blocos.

- A CÉLULA CMOS

Na figura 6, apresentamos o circuito básico de uma porta lógica CMOS. O princípio de funcionamento desse circuito consiste no fato de que em nenhum momento existe um caminho contínuo entre VCC e TERRA. Sempre que um dos Tp estiver acionado pelo menos um dos Tn estará cortado. Assim, a capacitância de carga C_g é conectada à TERRA ou VCC em função das entradas A e B. Como só existe fluxo de corrente nos momentos de carga e descarga o consumo estático é desprezível.

A configuração típica de uma célula de GA CMOS é de pelo menos dois pares de transistores complementares com grade, dreno e fonte flutuantes. Como uma característica dos circuitos MOS é o uso intensivo de portas de transmissão convém que pelo menos um dos pares tenha grade independente. Com essa configuração, podemos implementar inversores, portas lógicas NAND / NOR ou portas de transmissão. Na figura 7, temos alguns exemplos típicos de células CMOS (ref.12 e 18). Pode-se observar que são muito semelhantes entre si, divergindo apenas quanto ao número de pares de transistores e, quanto aos transistores de grade comum. Essas configurações geram normalmente arranjos em fileiras, mas também existem variações adaptadas para GA com arranjos mais compactos.

O projeto de circuitos com GA mantém muita semelhança com o projeto com CI padrão. Na verdade, esse é um dos principais fatores de sua popularidade. No entanto, a decisão de utilizar um GA, ou outro CI dedicado, deve resultar de uma análise cuidadosa das especificações funcionais e de desempenho do sistema e, de considerações de custo, confiabilidade e mercado. Como essa análise foge ao escopo desse trabalho, consideraremoa apenas que ela foi realizada e que, concluiu-se pela utilização de um GA para implementação de parte do sistema.

A seguir é preciso decidir pela opção tecnológica mais conveniente. Alguns dos principais critérios de escolha são:

. a frequência de trabalho do circuito (ligada ao atraso típico da porta);
. a potência máxima consumida e/ou dissipada ;
. as especificações de entrada e saída (níveis de corrente e tensão) ;
. faixa de tensão de alimentação ;
. nível de integração necessário (número de portas).

Na figura 8, mostramos um gráfico das faixas de integração x atraso típico de porta das principais famílias tecnológicas. Na Tabela 2, temos alguns dados típicos de cada uma dessas famílias (ref. 20 a 24).

Após ter determinado a opção tecnológica, mais conveniente, devemos definir que GA iremos utilizar. Além dos critérios de desempenho, supostamente atendidos, influenciam nessa decisão critérios bastante diversos, tais como: a variedade de opções de número de portas e E/S, a qualidade da biblioteca de funções, a disponibilidade de recursos de CAD e a disponibilidade do fornecedor.

Selecionado o GA, devemos gerar um esquema lógico adequado ao GA escolhido, utilizando as células disponíveis na Biblioteca de Funções. A seguir, o esquema resultante deve ser verificado. No projeto de um CI, a montagem do protótipo com CI padrão pode ajudar mas não é suficiente, dada a grande diferença de condições. O recurso mais comum é a utilização de programas simuladores. A simulação pode ser realizada em vários níveis . O primeiro é a simulação lógica simples, usada para verificar a coerência lógica do circuito, sem considerações de tempos. Se o circuito passar nessa primeira verificação, pode-se proceder a uma simulação lógica mais completa atribuindo atrasos convenientes aos blocos lógicos. Finalmente, para caminhos mais críticos pode-se realizar uma simulação mais detalhada , ao nível do circuito. É interessante observar que, nesse ponto a influência das interconexões no atraso de cada bloco ainda não é conhecida e de

ve portanto ser estimada. Após a realização do leiaute essas estimativas podem ser reavaliadas e, se necessário, a simulação pode ser refeita.

Estando o esquema lógico verificado e aprovado é preciso realizar então o leiaute da(s) máscara(s) de personalização. Nos primeiros GA, esse leiaute era feito manualmente com auxílio de folhas pré-impressas e decalques com as interligações das funções da biblioteca. Antes de iniciar o desenho das interconexões os decalques correspondentes às funções eram distribuídos convenientemente pelas células. Atualmente, o procedimento mais comum é entrar com a descrição do sistema lógico, em alguma linguagem adequada, em um sistema de PAC e, utilizar programas especiais para calcular a distribuição de função mais conveniente e, determinar o melhor caminho ou roteamento das interligações. É verdade que, normalmente esses programas não são 100% eficientes e muitas vezes deixam ligações incompletas e que, devem ser terminadas à mão. Mesmo quando não se tem esses programas ou, quando não se deseja utilizá-los, podemos fazer o posicionamento das células e roteamento manual através do computador, sem necessidade das folhas pré-impressas e decalques.

No caso do posicionamento e roteamento (PeR) automático não precisamos verificar o leiaute para nos certificarmos, de que ele corresponde verdadeiramente ao esquema lógico e, de que não foram violadas regras do leiaute.

No caso de PeR manual essa verificação precisa ser feita.

É preciso notar que os fornecedores/fabricantes de GA normalmente oferecem a possibilidade do cliente entregar o projeto em diversos pontos dessa sequência. Por exemplo:

. após a especificação funcional ou projeto lógico. Nesse caso o fornecedor do GA se responsabiliza pela adaptação do projeto ao GA e pela subsequente fabricação e testes ;

. após a realização do leiaute, que é então transmitido ao fornecedor em fita magnética ou através de linha de comunicação.

É claro que, quanto mais cedo na sequência entregar o projeto ao fornecedor maior será o custo. Por outro lado, menor será seu envolvimento com as atividades específicas do projeto de CI. Na figura 9, mostramos um roteiro das etapas de projeto com os pontos de transferência para o fornecedor do GA.

CARACTERÍSTICAS	USO GERAL	PROGRAMÁVEIS	DEDICADOS		
			SEMI-DEDICADOS		SOB-MEDIDA
			PRÉ-FABRICADOS	PRÉ-CARACTERIZADOS	
Flexibilidade de Personalização	Não existe	Limitada	boa	alta	total
Prazo de Desenvolvimento	-	Dias	1 a 4 meses	3 a 6 meses	1 a 2 anos
Custo de Desenvolvimento (K$)	-	-	5 a 40	30 a 90	50 a 500
Riscos de Fracasso	-	Muito pequeno	baixo	médio	alto
Facilidade de Alterações	-	Muito fácil, prazo e custo mínimos	Fácil com prazo e custo razoáveis	fácil com prazo e custo um pouco maiores	difícil, demorado e caro
Quantidades típicas	1 a 100000	1 a 100000	> 1000	> 20000	> 70000

Tabela 1 - Classificação e Características dos CI

TECNOLOGIA	NÚMERO DE PORTAS	NÚMERO DE ENTR./SAÍDAS	POTÊNCIA (mW/PORTA)	ATRASO (ns)	FREQUÊNCIA MÁXIMA (MHz)	COMPATIBILIDADE DE E/S
CMOS	200 - 10000	24 - 180	0.003-0.05/MH	1 - 10	10 - 100	MOS-TTL-LS
ECL	100 - 3000	24 - 180	2 - 5	0.3 - 1.5	200 - 700	ECL-LS
I2L	200 - 19000	30 - 140	0 - 1	10 - 50	2 - 20	LS- TTL
TTL	200 - 3000	30 - 150	1 - 20	2 - 6	20 - 80	TTL-LS

Tabela 2 - Parâmetros típicos de GAs nas principais tecnologias

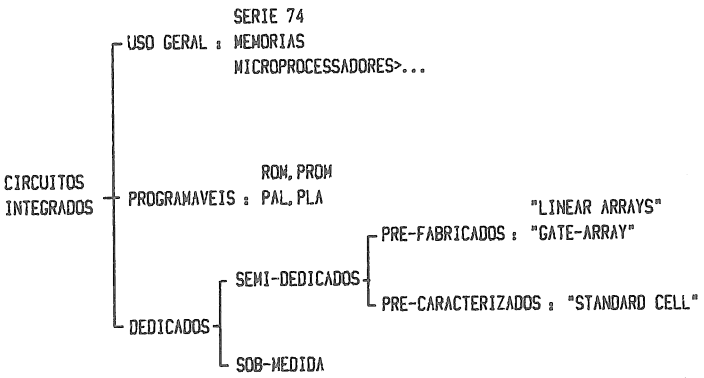


FIGURA 1.- CLASSIFICAÇÃO GERAL DOS CIRCUITOS INTEGRADOS

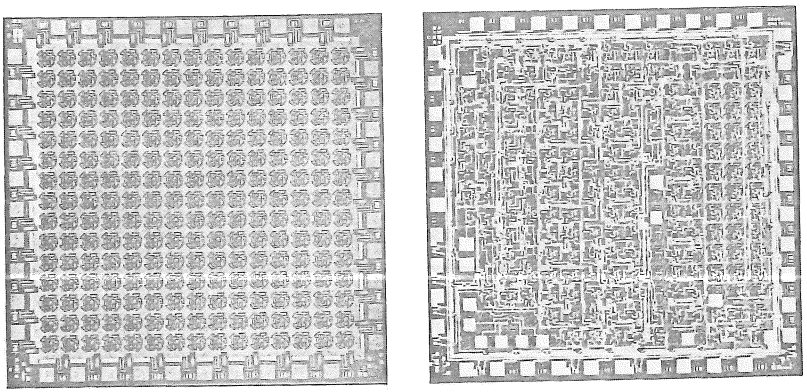
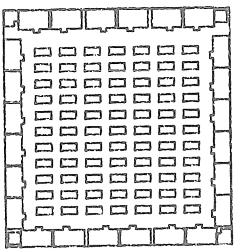


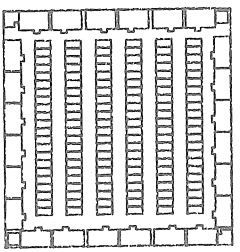
FIGURA 2.- UM "GATE-ARRAY" ANTES E DEPOIS DA PERSONALIZAÇÃO (ref. 6)

FIGURA 3

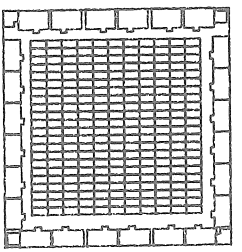
TIPOS DE ARRANJOS



A) EM BLOCOS



B) EM FILEIRAS



C) COMPACTO

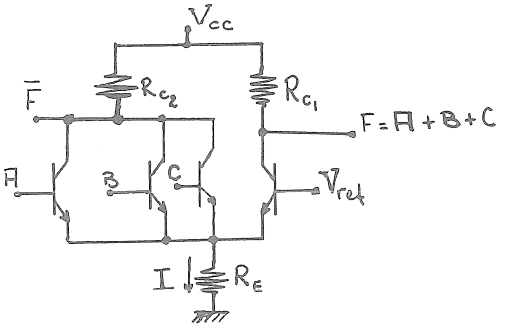


FIGURA 4.- CIRCUITO BASICO ECL

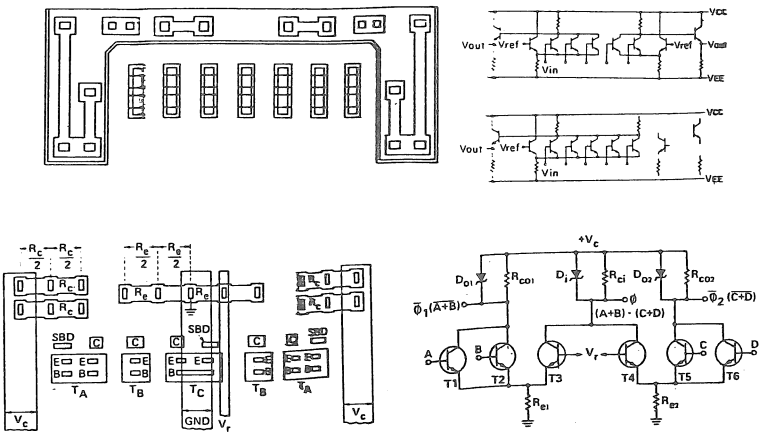


FIGURA 5.- EXEMPLOS DE LEIAUTE E CONFIGURAÇÃO DE CELULAS ECL (ref. 7-11)

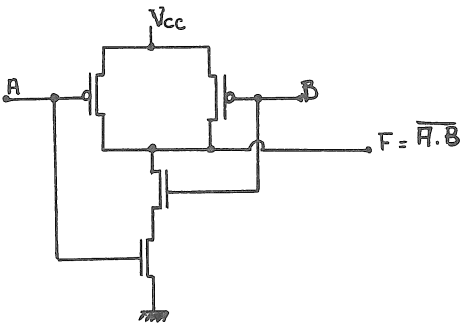


FIGURA 6.- CIRCUITO BASICO CMOS

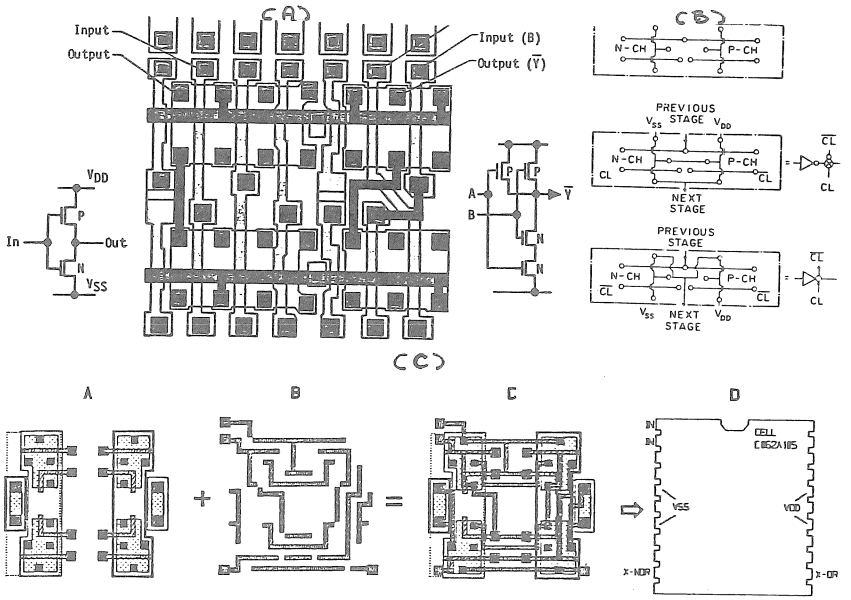
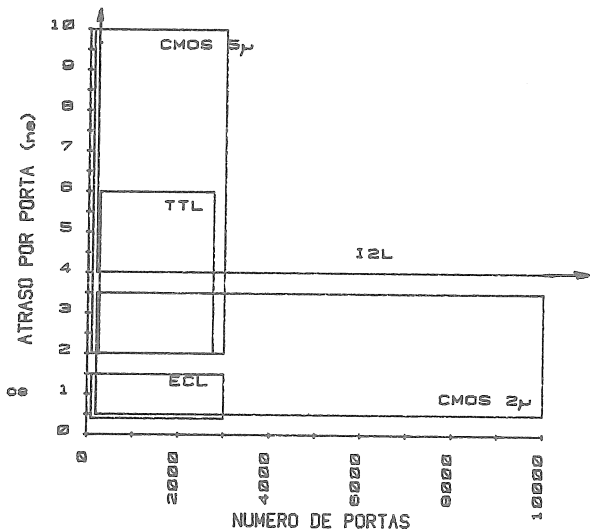


FIGURA 7.- EXEMPLOS DE LEIAUTE E CONFIGURAÇÃO DE CELULAS CMOS (ref. 12-24)

FIGURA 8.- GRAFICO DENSIDADE x ATRASO



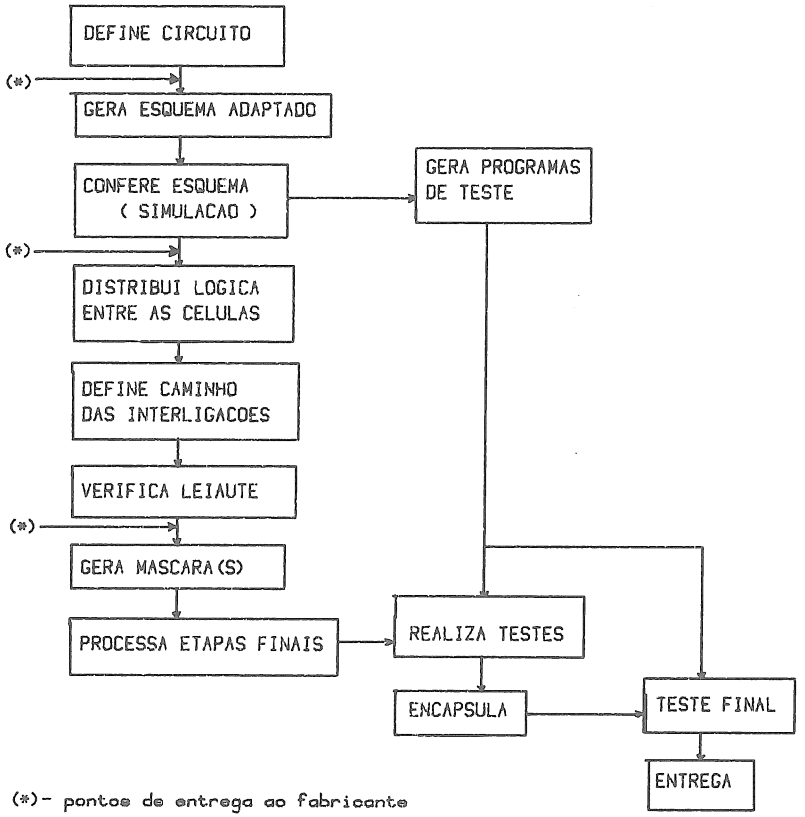


FIGURA 9.- ROTEIRO DE PROJETO COM GATE-ARRAYS

REFERÊNCIAS:

1. An Overview of Semi-Custom IC Technologies - Tutorial Session
P.Hicks
Second International Conference on Semi-Customs, London,

2. Custom-Semicustom Business Report
S.Z.Szirom
VLSI Design, Jan/Feb. 82.

3. Custom Electronics - Product Focus
Electronic Engineering, Jul. 82.

4. Semi-Custom Circuits Arrive
J.Bond
Computer Design, March 84.

5. Les Circuits a la Demande Envahissent tous les domains de l'electronique
Mesures, 18 fevrier 1985.

6. Changing Economics in Custom ICs - Special Report
Electronics Engineering , mid-may 78.

7. A 1500 Gate, Random Logic, Large-Scale Integrated (LSI)
Master Slice
R.J.Blemberg e S.Brenner
IEEE J.Solid State Circ., SC-14, nº 5, Oct.79.

8. A Bipolar 2500 Gate Subnanosecond Masterslice LSI
M.Nakaya et al
IEEE J.Solid State Circ., SC-16, nº 5, Oct.81.

9. A Subnanosecond 2000 Array with ECL 100K Compatibility
F.Sato et al
IEEE J.Solid State Circ., SC-19, Feb.84.

10. Projeto de um Gate-Array ECL de Lógica de Dois Níveis
M.Moraes
Dissertação de Mestrado, EPUSP, 83.

11. Projeto de um Gate-Array ECL de Lógica Empilhada
M.Friedel
Dissertação de Mestrado, EPUSP, 83.

12. Introduction to MOS LSI Design
J.Mavor, M.A.Jack e P.B.Denyer
Addison Wesley Pub.Co., 1983.

13. Designing with Gate-Arrays
AMI
Electronic Engineering, Mar/April/May-82.

14. CMOS/SOS Automated Universal Array
F.Borgini e B.Suskind
IEEE J.Solid State Circ., SC-16, Oct.81.

15. A CMOS/SOS Gate-Array with a New Customization Technique of Cutting
N.Sasaki e M.Nakano
IEEE Trans. Electron Devices, ED-29, Oct.82.

16. A 6K Gate CMOS Gate-Array
H.Tazo et al
IEEE J.Solid State Circ., SC-17, Oct.82.

17. A CMOS 5000 Gate-Array
J.Duhachek e W.N.Heikkila
Journal of Semi-Custom Ic, Vol.1 , nº 1, 83.

18. An Advanced ISO-HCMOS ULA of Semi-Custom Applications
F.Marquis e C.Groves
Journal of Semi-Custom IC, Vol.1 , nº 1, 83.

19. Uma Matriz de Portas C^2L
T.Costa e L.C. Molina Torres
IV Simpósio Brasileiro de Microeletrônica, São Paulo, 1984.

20. Projeto de Circuitos Integrados Semi-Dedicados CMOS
L.O. Fontenelle Gonçalves
IV Simpósio Brasileiro de Microeletrônica, São Paulo, 1984.

21. Changing Patterns in Customs ICs Alter Design Bondaries
D. Eidsmore
Computer Design, Feb. 83.

22. Design Guides to Gate-Arrays
D. Eidsmore
Digital Design, May 83.

23. EDN Semi-Custom IC Design Series
A.Rappaport
EDN, Sept.1, 1983

24. EDN Semi-Custom IC Directory
A. Rappaport
EDN, Feb. 17, 1983.
EDN. Feb. 23, 1984.

UMA METODOLOGIA DE PROJETO DE CIRCUITOS INTEGRADOS SEMI-DEDICADOS CMOS NA MODALIDADE GATE ARRAY

LUIS OTAVIO FONTENELLE GONÇALVES*

SUMARIO

Este trabalho tem por objetivo apresentar uma Metodologia de Projeto, que permita ao Engenheiro de Sistemas Brasileiro utilizar, de maneira confiável, técnicas modernas de Projeto de Circuitos Integrados Semi-Dedicados na modalidade Gate Array, com vistas ao desenvolvimento de Sistemas.

ABSTRACT

The purpose of this paper is to present a Design Methodology to be used by the Brasilian System Design Engineers, in order to permit them to reliably use the modern techniques of Semi-Custom IC Design using the Gate Array approach, in the development of their Systems.

* Engenheiro de Eletrônica (ITA, 1967)
Gerente de Projetos da Elebra Microeletrônica S.A.
Rua Bogaert, 326
Fone: PABX (011) 272-4255 CEP 04298

INTRODUÇÃO

Pesquisas recentes no mercado de usuários potenciais de circuitos integrados semi-dedicados nos Estados Unidos indicaram que, em 1984, 83% deste mercado ainda não tinha aderido à modalidade Gate Array, preferindo se ater aos componentes de catálogo. Para 1988, no entanto, as previsões são otimistas, e indicam que mais da metade dos usuários potenciais estarão utilizando Gate Arrays.

O principal elemento motivador dessa adesão em massa aos Gate Arrays foi o barateamento da produção e consequente diminuição do volume mínimo economico de encomenda. Hoje em dia pode-se pensar em encomendas de cerca de 1000 unidades, com custo por componente apenas cerca de 100% superior ao de uma encomenda de 50 000 unidades.

Isso permitiu que companhias com pequeno volume de produção passassem a empregar Gate Arrays.

Outro elemento motivador foi a facilidade de projeto. A rigor um projeto com Gate Arrays não exige da Engenharia de Sistemas um volume de conhecimento maior que aquele necessário para um projeto com componentes de catálogo. Com o aumento da densidade dos circuitos integrados, tornou-se possível a implementação de células com grau de complexidade bastante grande, como por exemplo uma memória 16x4 similar a 74LS189, ou uma unidade lógica - aritmética semelhante à 74LS181.

As principais vantagens de um sistema implementado com Gate Arrays sobre um sistema cuja lógica aleatória é implementada com componentes tipo 74LS são:

- diminuição do tamanho do sistema
- redução do custo dos componentes
- simplificação da montagem
- diminuição da potência consumida
- melhoria do desempenho
- maior segurança industrial

Este último item é particularmente importante para a Engenharia de Sistemas Brasileira, porque a engenharia reversa sobre circuitos dedicados pode exigir um volume de recursos maior que o próprio projeto.

Vemos então como bastante oportuna a implantação de facilidades de projeto de CI's semi-dedicados no Brasil, e entendemos que estas facilidades terão necessariamente de ser implantadas de modo a criar uma imagem de confiança nos clientes.

Para tanto é essencial uma Metodologia de Projeto, que defina de forma inequívoca a interface entre o cliente e o fornecedor.

A METODOLOGIA

Chamaremos de Fornecedor a entidade que irá fornecer ao Cliente as ferramentas necessárias ao desenvolvimento do projeto. O Fornecedor será também o responsável pelo

provimento dos componentes, caso o cliente decida pela produção do Sistema.

O relacionamento entre o Fornecedor e o Cliente poderá se fazer de duas maneiras:

a) O Cliente projeta o CI utilizando as ferramentas do Fornecedor.

b) O Fornecedor projeta o CI para o Cliente, a partir de especificações bem definidas.

A primeira opção é certamente a mais conveniente para ambas as partes, mas o Fornecedor tem que estar apto a desempenhar a segunda opção. A grande dificuldade da segunda é a definição da interface entre o cliente e o Fornecedor, como veremos em seguida.

Caso o Cliente esteja apto a desenvolver o seu próprio projeto, ele deve procurar o Fornecedor com as seguintes etapas já cumpridas:

- Especificação Funcional, incluindo diagrama de tempo dos sinais de entrada/saída.
- Projeto Lógico, o qual poderá estar desenvolvido a nível, por exemplo, da família 74LSTTL.

O Cliente será então submetido a um curso de uma semana de duração, no qual lhe serão apresentadas as ferramentas necessárias à conclusão do seu projeto. As etapas a serem vencidas, basicamente, são:

- Entrada do Circuito: O objetivo é gerar um programa que descreva as interconexões entre os blocos funcionais.

  Entende-se aqui por blocos funcionais aquelas funções lógicas constantes na biblioteca de funções do Fornecedor.

  Idealmente, esta biblioteca deve conter as funções constantes nas famílias lógicas mais conhecidas (74LSTTL, 4000CMOS, etc), podendo ir além, apresentando por exemplo geradores de sequências pseudo-aleatórias, contadores Johnson, etc.

  Nota-se aqui a aplicação do conceito do projeto hierarquizado em que, a partir de células primitivas, criam-se funções lógicas mais complexas, de modo a facilitar o entendimento do Sistema. Mais tarde, após terminada a entrada do circuito e verificada a sua consistência mediante uma compilação, a lógica será reduzida às células elementares.
- Verificação do Circuito: Após compilada a entrada do Circuito e verificada a sua consistência, e reduzida a sua lógica a simples células elementares, será rodado um programa que fornecerá, a partir da conectividade e da complexidade do circuito, uma estimativa do atraso para cada nó. Esta estimativa de atraso é estatística e baseada numa "cultura" que o Sistema de Desenvolvimento possui sobre o processo.
- Verificação Funcional: é a simulação lógica do circuito.

  Para tanto será gerado um programa que simulará as formas de ondas de entrada.

Este programa, juntamente com a entrada do circuito, será submetido a uma simulação lógica que terá como saída uma descrição no tempo do comportamento de todos os nós de circuito. O arquivo de saída pode ser especificado para validar funcionalmente o circuito, bem como para a deteção de "glitches" ou atrasos críticos de propagação em determinados ramos. A simulação lógica deverá efetuar uma cobertura total de circuito, ou seja, todos os nós deverão ser exercitados ao menos uma vez em ambos os sentidos (0→1 e 1→0), e o Sistema de Desenvolvimento deve estar apto a fornecer este dado.

- Geração do Vetor de Testes: Basicamente é um programa que transforma o arquivo de saída da simulação lógica e o formata de modo que as entradas passam a ser as entradas do equipamento automático de teste (ATE), e as saídas serão usadas pelo ATE para verificar o funcionamento do CI na produção.

- Diagrama de ligações: Após a verificação funcional já se pode definir exatamente o tipo de família a ser utilizada, em função dos atrasos verificados.

  O grau de complexidade do componente também é facilmente definido em função do número de portas equivalentes que o CI possui. O tipo de empacotamento é especificado pelo Cliente, e é rodado o programa que define as ligações entre o empacotamento e o "chip". É importante que estas informações sejam repassadas através de um software interativo, que alerte o CLiente sobre ligações proibidas, números de pinos de V_{dd} e V_{ss} necessários, conexões de substrato, etc.

Todas estas etapas a serem desenvolvidas pelo Cliente serão assistidas pelo Fornecedor. Para tanto será designado um Engenheiro de Aplicação, que será o responsável da parte do Fornecedor pelo Projeto. Note-se que, uma vez terminadas essas etapas, está automaticamente definida a interface entre Cliente e Fornecedor. Será assinado um documento por ambas as partes, em que o Cliente afirma que o componente por ele desejado possui as características constantes no diretório para ele designado quando do início do projeto, e o Fornecedor se compromete a fornecer uma certa quantidade de protótipos com aquelas características.

Caso o Cliente solicite o projeto ao Fornecedor, este deverá colocar o Cliente a par de todas as etapas a serem desenvolvidas e o mesmo documento deverá ser assinado. Aqui surgem dúvidas quanto à capacidade do Cliente de entender as saídas dos programas e identificá-las com as suas especificações, bem como quanto à capacidade do Fornecedor entender as especificações do Cliente, nem sempre suficientemente maduras.

Observa-se normalmente na Engenharia de Sistemas no Brasil uma tendência a se atropelar a especificações, em favor de um prazo menor para a conclusão do primeiro protótipo, o qual por sua vez muito pouco tem a ver com o protótipo industrial. Obviamente em projeto de circuitos integrados, dado o alto custo de qualquer retrabalho, essa é uma maneira de projetar que deve ser evitada.

Uma vez apresentadas as etapas da metodologia a serem vencidas com as ferramentas de software à disposição do Cliente, passemos a uma rápida descrição do que consideramos ser matéria indispensável do curso:

- Informação básica: entendemos como informação básica aquela necessária ao entendimento dos itens seguintes:

  Estrutura CMOS
  Estrutura de um Gate Array
  Conceito de células e funções
  Conceito de portas equivalentes

- Documentos de Especificaçao do Circuito e de Aceitação das Etapas: Estes documentos são divididos nos seguintes blocos:

  a) Especificações iniciais: informações sobre o usuário, formato de entrada/saída, caminhos críticos, limitações de potência, etc.

  b) Relação dos arquivos para aceitação do projeto.

  c) Especificações Finais: adiciona às especificações iniciais a descrição do circuito e os resultados da simulação.

  d) Documentos Finais: são 3, a saber:

  1. Relatório de Término do Projeto, que libera o projeto para lay-out.
  2. Relatório de Aprovação de Desempenho, que verifica o projeto com os comprimentos reais das interligações, e que libera o projeto para fabricação.
  3. Relatório de Aprovação do Protótipo, que libera o Protótipo para Produção, a ser coberta em contrato separado.

- Escolha do Tipo de Gate Array: O projetista, para escolher a família de Gate Array a ser utilizada no projeto, precisa simular o seu circuito com dados de atraso da família que ele ainda não escolheu. É o chamado "dilema do Projetista", que só a experiência soluciona. No entanto, devem ser apresentados ao projetista os itens que o ajudam a resolver esse dilema; quais sejam:

  a) Como elaborar um diagrama de blocos funcional a partir da Biblioteca de Funções
  b) Como identificar os caminhos críticos.

- Escolha do Tamanho do Gate Array: Que tamanho deve ter o componente é outro item importante, e que vai ditar em boa parte o custo de produção.

- Otimização do Projeto: Um Gate Array pode ser otimizado de 3 maneiras:

  a) Melhorando-se a velocidade
  b) Reduzindo-se o número de portas
  c) Reduzindo-se a pinagem

  As técnicas para se atingir esses objetivos devem ser discutidas.

- Seleção do Tipo de Empacotamento: Muito embora o Cliente normalmente já saiba que tipo de empacotamento ele quer, ele deve ser informado sobre as facilidades de empacotamento à sua disposição.

- Preparação do Diagrama de Ligação: Como vimos, é uma etapa importante do projeto, e como tal deve ser tratada no curso.

- Estimativa de Potência: Importante em algumas aplicações, a estimativa de potência é um dado bastante fácil de ser levantado em Gate Arrays, dado que uma família consolidada deve ter uma figura de dissipação, expressa em (mW) / (frequência) x (número de portas equivalentes).

- Testabilidade e Confiabilidade: Com o aumento da complexidade, a questão da testabilidade assumiu uma enorme importância, e na verdade ela constitui item de aceitação do projeto. Normalmente não deve ser aceito um padrão de testes que não cubra todos os nós do circuito. A confiabilidade também deverá ser abordada, recomendando-se por exemplo o emprego de sistemas síncronos.

- Preparação do Padrão de Testes: O padrão de testes a ser usado na produção será aquele fornecido pelo Cliente, derivado da simulação lógica. Porém os fornecedores costumam estabelecer um limite para o comprimento do padrão de testes.

Especial atenção deve ser dada ao diálogo homem-computador, já que o engenheiro de sistemas, em geral, não está familiarizado com o uso de terminais.

Ele deverá ser inicializado ao acesso ao computador, bem como à edição de arquivos. Um software interativo é essencial para que o engenheiro de sistemas encare o computador de forma amigável.

O curso não pode terminar sem um exemplo de aplicação, em que o Cliente desenvolve um projeto simples, fazendo uso do software e vencendo todas as etapas acima descritas.

A. A. SUZIM

SUMÁRIO

Este artigo apresenta o trabalho que vem sendo feito para construir um montador automático de blocos funcionais para concepção de circuitos integrados. Primeiramente um modelo geral do circuito é apresentado e, em seguida são definidos os blocos funcionais para os quais são programados os procedimentos geradores. A nova estratégia consiste em encapsular a descrição (gráfica e elétrica) das células numa subrotina geratriz que é chamada pelo usuário. O problema da parametrização das células, difícil de resolver no sistema gráfico pode ser facilmente resolvido pela passagem de opções do usuário através dos parâmetros da subrotina. Isto é comparado com um trabalho anterior que usava o conceito de biblioteca de células.

ABSTRACT

This paper presents the work been done to construct an automatic block assembler for integrated circuits design. Firstly a general model of the circcuit is presented then some functional blocks are defined to which building procedures are programmed. The new strategy consists of encapsulating the cells' description (graphical and electrical) in generating procedure called by the user. The problem of parameterisation of cells, difficult to solve in a graphical environment can easily be solved by user options passed as procedure parameters. This is compared with a former work that used the cell library concept.

Curso de Pós-Graduação em Ciência da Computação
Universidade Federal do Rio Grande do Sul
90.000 - PORTO ALEGRE - RS

I - INTRODUÇÃO

A necessidade de ferramentas adequadas para a realização de circuitos integrados complexos tem sido objeto de uma grande quantidade de estudos, trabalhos e publicações nos últimos tempos. Os investimentos que os governos e as indústrias fazem atualmente na área de microeletrônica mostra que a sociedade está apostando nesta nova tecnologia. De fato, as implicações sociais da informatização (sobre as condições de trabalho, modo de vida, nível de emprego, etc.) não podem deixar ninguém indiferente. E a informatização está alicerçada sobre a microeletrônica.

As ferramentas adequadas à realização de circuitos integrados complexos estão baseadas em metodologias que tentam automatizar o mais possível a concepção dos circuitos: partindo de uma descrição do problema utilizando conceitos de uma linguagem de alto nível, fases sucessivas de refinamento são aplicadas , decompondo-se os conceitos complexos em elementos cada vez mais simples até chegar ao nível dos elementos materiais. Quando do desenho dos primeiros circuitos integrados, um trabalho artesanal era satisfatório; o degrau seguinte da complexidade pode ser superado apenas com recursos computacionais gráficos. Hoje o trabalho se concentra na pesquisa de novas metodologias.

A Microeletrônica é uma atividade multidisciplinar. Hoje ela incorpora novas disciplinas desde a informática e sistemas até a área de equipamentos como laser, plasma, raios X, etc. Em consequência investimentos cada vez maiores são necessários (sempre contando com um retorno compensador).

Em um trabalho anterior [SUZ 81a, SUZ81b] uma metodologia de geração de partes operativas de circuitos integrados NMOS, assim como um conjunto inicial de células e o protótipo do programa de montagem haviam sido apresentados. A experiência com aquele sistema permitiu validar a metodologia e detectar alguns problemas práticos, como a proliferação de pequenas células. Um conjunto muito grande de células é difícil de manter, de caracterizar e, certamente, de usar. As células armazenadas em uma biblioteca são rígidas, pois o nosso sistema gráfico não permite parametrização.

Este sitema utiliza uma linguagem de programação de alto nível (le-lisp, no caso). As descrições das células são sequênciais de retângulos codificados dentro do próprio programa. Saõ em geral constantes, exceto os pontos que o projetista da célula pretende deixar como parametráveis. Estes serão calculados em tempo de execução (variáveis ou expressões do programa).

Um fato novo é que o projetista da célula deverá desenhá-la, descrevê-la e programar o algoritmo de montagem, de tal forma que o usuário veja apenas a função e não se preocupe com a montagem mas apenas com a especificação.

Uma característica de Lisp que facilita este tipo de trabalho é a possibilidade para tra

tar os parâmetros muitas vezes em quantidade variável, transmitidos dentro de uma lista.

Ao mesmo tempo em que esta estratégia é desenvolvida, o trabalho é utilizado para a definição de um conjunto de blocos funcionais na tecnologia CMOS porta de silício e dois níveis de metal. Além dos aspectos topológicos, com características bem diferentes do primeiro trabalho (NMOS), aspectos do comportamento elétrico são tratados.

Finalmente, a essência do problema de concepção de um circuito é a passagem da descrição comportametal de um sistema para o domínio físico geométrico |CON84|.

II - MODELO DO CIRCUITO

Ao se propor uma construção automática dos circuitos tem-se em mente um modelo de máquina que supostamente satisfaz a uma classe de problemas. Com esse modelo em vista são propostas primitivas que permitam descrever uma solução para o problema. A descrição do problema continuará sendo uma operação delicada, uma vez que exige um perfeito conhecimento do problema e da semântica das primitivas.

O modelo proposto nas referências citadas apresenta o circuito como composto de dois grandes blocos: bloco de controle e bloco operacional. O primeiro é responsável pelo sequenciamento das operações e o segundo, pela execução. A comunicação entre os blocos se faz por meio de um interface responsável pela adaptação de níveis elétricos, pela conexão e, algumas vezes, pela introdução de tempos (fases). Os sinais que fluem da parte (ou bloco) de controle (PC) para a parte operativa (PO) são chamados em geral de comandos e os que fluem no sentido inverso, de predicados. Existem situaçãoes particulares em que essa denominação pode parecer não muito adequada. Microprocessadores do tipo bit-slice (como a série 2900) são exemplos sugestivos da divisão entre PO e PC. Não se deve por outro lado associar o conceito de microprogramação com o modelo proposto mesmo que não raro os dois conceitos casem muito bem. (Fig.1)

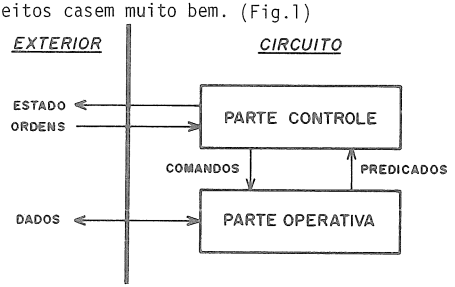


Figura 1. Modelo PC-PO de um circuito ou sistema

Vale ainda mencionar que o modelo se adapta ao conceito de hierarquização da descrição. Em um nível qualquer de trabalho, podem ser usadas primitivas cuja implementação direta seja inviável, sabendo-se que numa fase posterior essa mesma primitiva será detalhada (interpretada). Além disso é possível que uma parte de controle controle mais do que uma parte de operativa e, ainda, que uma parte operativa seja ela mesma uma máquina e, conse-

quentemente, decomponível (por resultado de interpretação) em uma parte controle e uma parte operativa, etc.

III - PARTE OPERATIVA

A parte operativa é caracterizada por conter os elementos de memorização, transformação e comunicação de informações, respectivamente memórias, operadores e barramentos. A necessidade das memórias e dos operadores é evidente. Os barramentos são em geral empregados nos circuitos mais complexos mas não são essenciais à PO. A faculdade de comunicar ou transportar os dados é importante mas pode ser feita sem barramentos.

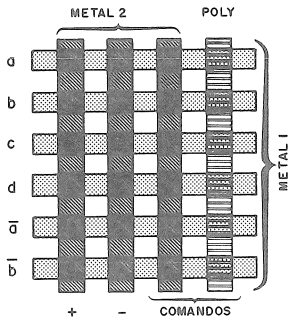
3.1 Os Barramentos

Utiliza-se barramentos quando é proibitivo ou inútil conectar diretamente todos os elementos que devem comunicar entre si. Vendo de outro lado, os barramentos são canais de comunicação partilhados no tempo e sugerem Partes Operativas bem estruturadas construídas ao redor de um sistema de protocolo bem definido. No caso de circuitos integrados, mesmos as características geo-métricas são fixadas (normalizadas). O conjunto de características do barramento define o sistema de barramento (junto a uma família de células). A escolha de um sistema de barramento é, pois, algo bastante delicado. Obviamente existem células que são mais ou menos comprometidas com o sistema de barramento, ou seja, algumas células só operam com um determinado sistema de barramento (sequência de tempos ou níveis lógicos) enquanto outras podem admitir mais de uma forma de operação.

Entre as células mais sensíveis ao tipo de barramento estão as memórias e, muitas vezes, são exatamente essas as primeiras células projetadas dando origem ao sistema de barramento. A primeira proposição, tendo em vista a existência de dois níveisde metal era de reduzir ao máximo a altura do barramento, por exemplo, levando a alimentação em metal dois e deixando o metal 1 para os dados (o interesse de se trabalhar com dois barramentos bifilares foi apresentado nas referências). O fato de ser CMOS sugeriu um re-estudo inclusive da utilidade de haver pré-carga e barramento complementar. Por tentativa se chegou a conclusão que a situação ótima para uma parte operativa que tenha um percentual significativo de sua área ocupado por circuito de memória é a mostrada na figura 2.

É praticamente impossível desenhar uma célula de memória CMOS com as regras de projeto disponíveis dentro da altura de seis passos de metal, mantendo uma forma interessante para a montagem do bloco de memória. Além disso a grande quantidade de interconexões necessárias em CMOS sugeriu a reserva de um dos níveis de metal para conexões locais e, se necessário, para controles, e o outro nível para os dados, alimentação e barramentos reserva. O sistema de barramento resultante é apresentado na figura 3.

O comportamento elétrico do barramento e as características elétricas da memória são mostrados na figura 4.



VERSÃO INICIAL

Figura 2. Barramento da P.O. CMOS

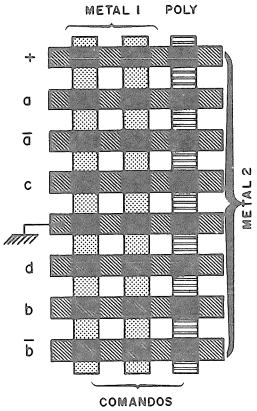


Figura 3. Forma Final do Barramento

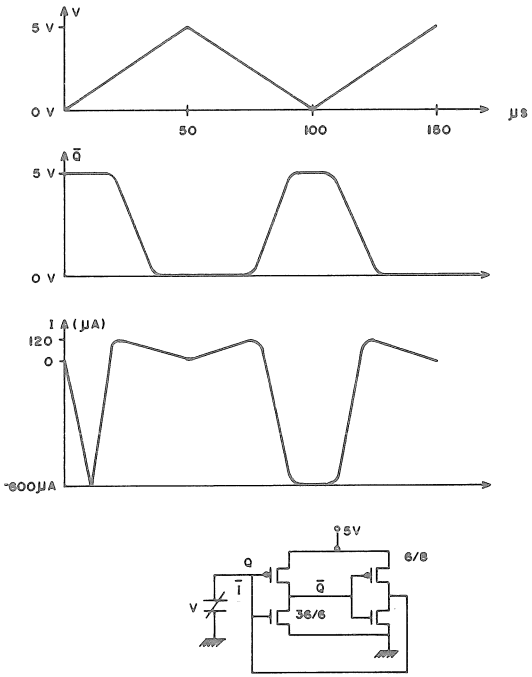


Figura 4. Características da Memória

3.2 As Memórias

Dos dois tipos de memória (ROM e RAM) o primeiro em geral é fácil de implantar. Quanto às memórias de leitura e escrita podemos ainda distinguir entre as memórias propriamente ditas que são pensadas para constituírem blocos de memória - caso em que conta muito a otimização de superfície e consumo elétrico - e as memórias isoladas constituindo em geral um registrador especializado onde conta a maleabilidade para adaptar-se a diferentes situações topológicas ou dos sinais de controle. Esses comentários valem obviamente para o contexto deste trabalho ou seja para o desenvolvimento da parte operativa do circuito.

As memórias de leitura/escrita destinadas a "uso geral" dentro dos circuitos podem ser colocadas em bloco se não houver conflito de acesso. Como mencionado acima buscar-se-á otimização em área e consumo. Para NMOS, o esquema da memória com dois inversores realimentados é uma solução perfeitamente adaptada. Tem-se um ponto a seis transistores (ou quatro transistores e dois resistores) e o elemento de carga do inversor tem função apenas de compensar a corrente de fuga. No momento da seleção em escrita no ponto de memória podemos imaginar um circuito equivalente a um NÃOOU. Em CMOS o comportamento é diferente já que não existe o elemento de carga:tudo se passa como se as funções fossem calculadas duas vezes e para cada porta lógica de n variáveis, 2xn transistores são utilizados. Jogando com as características geométricas dos transistores é possível trabalhar com a mesma célula como em NMOS (6 transistores). Naturalmente uma das grandes características do CMOS fica comprometida: seu baixo consumo. Técnicas como chaveamento da fonte de alimentação podem ser utilizados mas não serão tratadas aqui (figura 4).

Outros pontos de memória necessários na parte operativa do circuito muitas vezes exigem que se guarde uma informação gerada por uma função (incrementador, por exemplo) cuja saída, se se desejasse complementar, seria necessário acrescentar mais lógica.

Um circuito interessante, apesar de exigir o comando de escrita e seu complemento, é mostrado na figura 5.

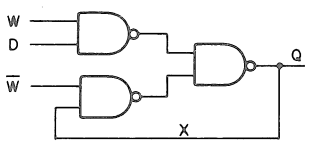


Figura 5. FF D Usado nos Registradores

O circuito é na realidade um multiplexador de duas entradas e os estudantes que desenharam uma PO CMOS utilizaram esta mesma célula para outros elementos como, por exemplo, para realizar lógica; pelo fato de ser utilizada para finalidades diversas a célula foi também chamada de célula mãe.

Essa mesma célula é utilizada como multiplexador simplesmente eliminando o laço de realimentação.

3.3 Os Operadores

Os operadores são circuitos combinacionais que realizam uma função de transformação sobre um ou mais operadores. São funções típicas: incremento, soma, deslocamento, ou, etc. Alguns operadores podem realizar várias funções e um parâmetro escolhe qual delas deve ser realizada num dado momento (caso típico das Unidades Aritméticas e Lógicas). Existem também operadores que podem realizar mais do que uma função ao mesmo tempo: são os circuitos de múltiplas saídas.

Construídos sobre um sistema de barramento, os operadores de uma Parte Operativa tem acesso fácil aos operandos; não há necessidade de "routing". As questões que se colocam são quanto ao tempo necessário para as operações, quanto ao número de registradores e se estes estão na entrada ou na saída do operador e principalmente, quais as operações que determinado operador deve realizar.

Sobre este último item, é conhecido que existe uma relação entre complexidade e área. Portanto, não há interesse em se dispor do mais complexo operador capaz de realizar um conjunto completo de operações. É necessário que se possa construir o menor operador possível capaz de realizar as operações necessárias para uma determinada aplicação. A dificuldade maior está em construir automáticamente este operador sabendo que cada aplicação tem seu conjunto de operações. Ocasiões há em que se pode implantar uma Unidade Aritmética e Lógica (UAL) Completa, outras há em que se escolhe um operador simples e acrescenta-se outro, feito especialmente para o circuito uma vez que a função não é frequentemente utilizada. Os operadores que estão ligados ao mesmo barramento podem trabalhar simultaneamente.

4. Montagem

No "lay-out" da memória apresentado na figura 4, aparecem dois bits de duas palavras contíguas e também as conexões com os barramentos. Esses desenhos foram feitos para uma minimização da utilização da área. Entre o desenho da célula básica e a montagem de um bloco de memória existe bastante trabalho. Este trabalho pode ser feito automáticamente desde que se programe um procedimento que permita qualquer combinação de elementos de memória. Por exemplo montar um número ímpar de palavras ou iniciar com um padrão especificado como: a primeira célula não deve ter contatos com o barramento, ou o contrário, deve

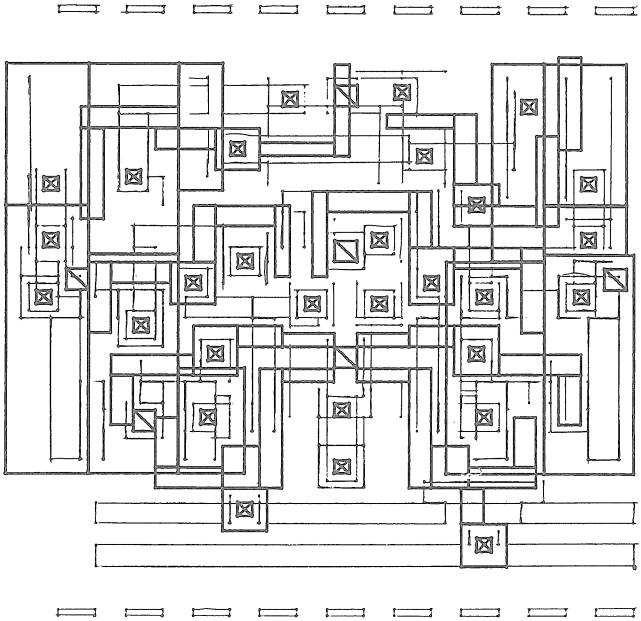


Figura 6. Multiplexador

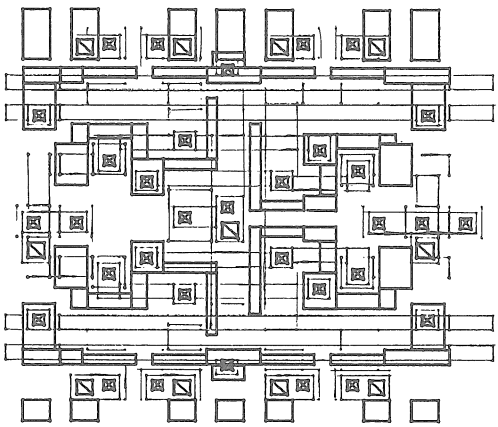


Figura 7. Memória

tê-lo.

A descrição de um bloco pode ser extensa e existem algoritmos cuja complexidade exige um trabalho grande. Como exemplo do primeiro caso podemos citar a unidade aritmética e lógica (incluindo o circuito de propagação de transbordo) e como exemplo da segunda classe de problemas pode-se mencionar os programas geradores de PLA, (incluindo-se os algoritmos de otimização lógica e topológica).

A descrição pode ter uma forma como o que segue (em Lisp):

```
(Setq l1 '(
     (5 12 4 7 d)
     (5 27 4 7 d)
     ...
```

onde os números entre parêntesis definem um retângulo e d é uma variável - o nível no caso. As variáveis podem corresponder a dimensões dos retângulos ou a seu posionamento. É com esse sentido limitado que se entende a parametrização das células. Conclui-se que:

- Todas as formas possíveis de uma célula devem ser previstas por quem a projeta.

- A parametrização deve ser minimizada - só será possível alterar dentro de um campo limitado de valores e

- O programa de montagem deve fazer a consistência desses valores.

Existem casos em que é preferível codificar não numa lista de retângulos, mas em listas de cada uma das coordenadas - x ou y - e das dimensões dos mesmos. No caso de elementos (retângulos), de mesmo tamanho, a lista pode conter apenas um elemento. O próximo elemento de uma lista é dado por:

```
(de prox (l1)
  (cond ((cdr l1)) (l1))) .
```

Embora pareça evidente que este método de trabalho é interessante, só a experiência poderá dizer quais são as limitações, os casos típicos e as excessões. É até possível que o modelo deva ser refinado, quiçá expandido.

Finalmente, é importante salientar que, uma vez caracterizado um bloco, todas as informações pertinentes podem ser integradas na própria descrição. É assim que os níveis superiores em abrangência poderão inquirir sobre propriedades dos blocos funcionais. Um sistema de avaliação de desempenho, por exemplos, poderá obter de cada bloco o tempo de atraso desse bloco, fornecidos os parâmetros necessários. A modelização é, nesse sentido, um trabalho iterativo, pelo menos na fase inicial. A hierarquia criada admite ramificações e agregações. Este trabalho pretende caracterizar o nível mais baixo da hierarquia: a descrição dos objetos mais elementares sob o ponto de vista funcional.

CONCLUSÃO

Na tentativa de harmonizar as descrições das células dos circuitos integrados com os níveis de descrição mais altos e de disciplinar seu armazenamento e montagem, sugere-se que essa descrição contenha não só a listagem dos elementos geométricos mas também o algoritmo de montagem. Algo como um código genético que permite montar a estrutura. Não só as informações geométricas são gerenciadas, mas também as informações elétricas como consumo, retardo, impedância, etc. O sistema deve ter capacidade de estrapolar para o bloco funcional as características das células. Isso implica inclusive em criar modelos para um conjunto de células (bloco). Todo esse conjunto de dados e algoritmos está dentro de um programa especializado em gerar um determinado bloco funcional.

A etapa seguinte será dedicada à elaboração das operações entre esses objetos. Uma linguagem de montagem de células já foi definifa na referência [SUZ 84b]. Uma evolução desta linguagem que permita a utilização de parâmetros para a definição de uma célula e, por outro lado, recupere informações desta mesma célula é, certamente muito interessante e difícil.

AGRADECIMENTOS

Gostaríamos de agradecer no Prof. Bernard Courtois e à sua equipe, em particular e Mohamed Jerraya. Num curto espaço de tempo foi possível iniciar a organização destas idéias que nos perseguiam há tempo. O intercâmbio faz parte do convênio CAPES-COFECUB.

Esse trabalho se situa no contexto do laboratório de microeletrônica do PGCC/UFRGS apoiado por Convênio FINEP e por bolsas e auxílios individuais de pesquisa do CNPq.

BIBLIOGRAFIA

[SUZ 81a] SUZIM Altamiro Amadeu "Data Processing Section for Microprocessor - "Like Integrated Circuits" IEEE Journal of Solid-State Circuits Vol. SC-16 nº 3 JUN/81 pp. 233-235

[SUZ 81b] SUZIM, Altamiro Amadeu "Etude des Pantres Operatives a Elements Modulaires para Processeurs Monolithiques. Thèse Docteur-Ingénieur INP - Grenoble - NOV 1981.

[CON 84] CONNIE U. Smith and DALLEN John A. Future Directions for VLSI and software Engineering. in VLSI ENGINEERING Edited by G. GOOS and J, HARTMANIS. Springer-Verlag 1984.

BRUNE, Osmar; CALAZANS, Ney; DILENBURG, Renato & PINHEIRO, Dércio. Parte Operativa CMOS. Relatório em preparação.

R. CAMPOSANO*, R. WEBER**

ABSTRACT

This contribution presents the main features of DSL (Digital Systems Specification Language) and its compilation into an internal form. DSL allows imperative and applicative specifications, concurrency at the operation and at the process level and the specification of design constraints such as time, area and power consumption. The internal form includes a first approach to a data-path design, the predecessor-successor relationship of the operations and a timing constraints graph. An example and some performance measures of the current implementation of the compiler conclude the paper.

RESUMEN

Esta contribución presenta las principales características de DSL (Digital Systems Specification Language) y su compilación a una forma interna. DSL permite especificaciones aplicativas e imperativas, paralelismo al nivel de operaciones y al nivel de procesos y la especificación de condiciones de borde como velocidad y área. La forma interna incluye una primera aproximación al diseño de la parte de datos, la relación predecesor-sucesor entre las operaciones y un grafo de restricciones de tiempo. El trabajo finaliza con un ejemplo y algunos datos sobre el desempeño de la actual implementación del compilador.

* Forschungszentrum Informatik and ** Institut für Informatik IV (Prof. D. Schmid), Universität Karlsruhe, Haid- und Neustr. 10-14, 75 Karlsruhe 1, F.R.Germany. This work was partially supported by the BMFT (Bundesministerium für Forschung und Technologie) and the SIEMENS AG under grant NT28093.

Digital systems can be described at different levels. We distinguish ideally the behavioural, the structural and the geometrical levels (see also [Shiv83], [CKR84b]). At the behavioural level a system is described in terms of its operations, at the structural level the system is given as the interconnection of modules, and at the geometrical level the exact layout of the masks is specified. Examples are DAISY [John83], ISPS [BBCS79], MIMOLA [Marw84] and DSL [Rose82] at the behavioural level, Zeus [Lieb84b], DDL [DuDi68] and STRUDEL [CaTr84] at the structural level and CIF [MeCo80] at the geometrical level. A bibliography of the numerous existing hardware description languages is given in [Nash84]. Standardization efforts are reviewed in [Waxm84].

The work reported in this paper is part of a larger project which aims at synthesizing IC's from 'pure' behavioural level descriptions. First they are transformed into (also 'pure') structural descriptions which are then converted by a placement and routing system into a geometry (Fig. 1). We deal mainly with the first step, i.e. the transformation of a behavioural description into a structural one. For the transformation of structures into chip layouts we use for example the layout system VENUS [GHHS84] developed by Siemens for automatic placement and routing of CMOS standard cells.

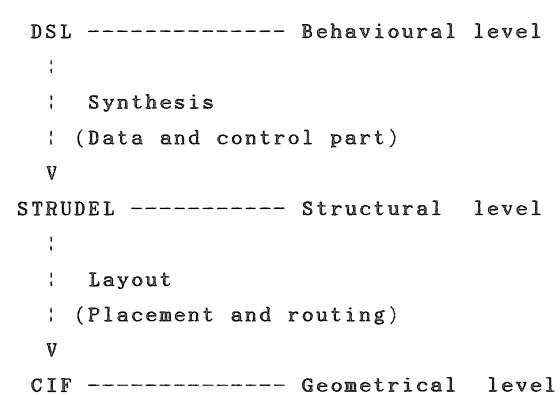


Figure 1. The DSL synthesis approach

There have been reported several approaches to the synthesis of digital systems from behavioural level specifications, mainly 'direct' compilation [DPST81], transformations using graph grammars [GiKn84], data flow analysis [Rose84] and expert systems [KoTh83b]. While some results reported suggest that synthesis (in the above sense) yields reasonable circuits, there is strong evidence that none of these approaches alone can claim to be general, i.e. to be feasable for a large set of designs.

Because of this lack of 'general' techniques, we separate synthesis into different steps, some of them algorithmic and some of them not. The compilation of a behavioural level specification into a first, not at all optimized design of the data path structure and the control sequencing is our main topic in this paper. This occurs in a pure algorithmic way (e.g. no 'artificial intelligence'), leaving perhaps more 'intelligent' activities for later steps.

The language presented in this paper evolved from former versions reported in [CaRo80], [Rose82], [CKR84a], [Rose84]. The additions and changes yielded, even if the name remained the same, a significantly different language. A detailed description of the language and the internal form can be found in [CaWe84], [CaWe85]. In the following we discuss this language and its distinctive features (chapter 2). The internal form into which DSL is compiled defines (only partially as we will see) the semantics (chapter 3). The compiler is presented with some emphasis on the performed consistency checks and some performance measures of our actual implementation (chapter 4). We conclude with a brief summary and some open questions (chapter 5).

2. THE DIGITAL SYSTEMS SPECIFICATION LANGUAGE DSL

To specify digital systems at the behavioural level several languages have been used. The definition of a specific language offers the advantage that the language can be designed according to the special needs of the application. Another approach consists in using an existing programming language such as

PASCAL or ADA [GiKn84] with the obvious advantages. Our main interest in defining DSL was to provide the means for a 'pure' behavioural specification, including timing and electrical behaviour to some extent. The power and comfort of modern programming languages should be given. The main problems we had do deal with were

- what style of specification, i.e. imperative or applicative should be used,
- how to specify concurrency,
- how to allow modularity and hierarchies,
- how to include the description of timing and electrical behaviour and
- which design constraints should be included.

In general DSL shows some similarity with PASCAL. The control structures are the same, i.e. IF-THEN-ELSE, CASE, WHILE-DO, DO-UNTIL and FOR-DO. The data structures are restricted to ARRAYS, while the basic data types are LOGICAL, ONE_COMP, TWO_COMP, BCD, FIXED and FLOAT. The allowed operators include the usual logical, arithmetic and relational operators. It should be noticed that the choice of basic data types and operators is reflected in the basic hardware modules that must be available, e.g. floating point division, two's complement addition, etc. Of course these modules can also be built by a synthesizer from simpler primitives.

A DSL program may include one applicative part and as many imperative processes as necessary. Global actions not constrained to a certain point in time, such as resets, interrupts, etc. are naturally specified in the applicative part. Sequential algorithms, the behaviour of finite automata and sequential behaviour in general are often more comfortably described with imperative procedures.

Concurrency can be specified in DSL at the operation and at the process level. Concurrent operations must be separated by commas inside a FORK-JOIN construct. FORK (JOIN) can be abbreviated by [(]). To allow control over concurrent processes, control operations nre provided. They are RESET, START, STOP and CONTINUE and have the following meaning:

OPERATION	DATA PART	CONTROL PART	ACTION
RESET	INITIAL STATE	INITIAL STATE	STOP EXECUTION
STOP	SAME STATE	SAME STATE	STOP EXECUTION
START	SAME STATE	INITIAL STATE	START EXECUTION
CONTINUE	SAME STATE	SAME STATE	START EXECUTION

Figure 2. Imperative Procedure control operations in DSL

The control operations can be applied only to imperative (sequential) procedures and allow, inside a FORK-JOIN, the specification of concurrent processes.

Modularity in DSL is partially achieved by permitting as many imperative processes as necessary. Moreover, a powerful abstraction mechanism permits also hierarchical specifications. The functions performed by a DSL program are (optionally) specified in the PERFORMED FUNCTION statement. Each function is characterized by its name, the inputs (interface, pins) it uses as arguments, the generated outputs (interface, pins) and the control signal sequence that must be applied to the remaining circuit interface to perform exactly this function. The name of the function can be used as an operator in other DSL programs that can also be compiled separately.

The electrical behaviour can be specified only globally and at the interface, e.g. FANIN and FANOUT, VOLTAGE LEVEL, POWER, etc. It is used to select the technology, the interface drivers and as a global constraint for the synthesis process (e.g. power consumption). Timing can also be specified as a global constraint (FREQUENCY). Moreover, it is possible to specify the

delay of any operation or group of operations in absolute time or in clock cycles. This feature is intended to impose timing restrictions to the synthesis, thus allowing to specify the delay in critical paths or other portions of a design.

Besides synthesis the other important application of DSL is automatic test pattern generation and design for testability. We are currently developing two approaches. Roughly the first one consists of the random test pattern generation for the combinational logic using a probabilistic method to estimate the fault coverage and making the storage elements accessible by a scan path or BILBO's [Wund84], [KuWu84]. The second approach attemps to generate test patterns already at the behavioural level by imposing certain restrictions to the allowed operations.

Figure 3 shows an example of a DSL program for calculating out=exp(in) using the Taylor series expansion.

```
   CIRCUIT exponentiation;
(* sequential circuit that calculates output=e**(input) *)
   INTERFACE
           vcc            : 12V;
           gnd            : GND;
           input(15..0)   : INPUT FANIN 1;
           output(15..0)  : OUTPUT FANOUT 10;
           enable         : INPUT FANIN 1;
           clk            : CLOCK FANIN 1;
   POWER         100 mW;
   VOLTAGE       12.00  V;
   HIGH LEVEL    11.00  V;
   LOW  LEVEL     1.00 V;
   TECHNOLOGY    CMOS;
   AREA          30 mm2;
   FREQUENCY 0 TO 500 kHz;

   PERFORMED FUNCTION
           output:=#exp(input)
           CONTROL
              (enable:='0' CYCLES=1);
              (enable:='1' CYCLES=48)
              END;
```

```
VAR     x,y : FIXED (8,8);
        i   : LOGICAL (4..0);
CLOCKBASE clk;

  APPLICATIVE
     output:=y,
     IF enable=0 THEN x:=1,y:=1
                 ELSE START calc
     FI
  END APPLICATIVE;

  IMPERATIVE calc;
     (FOR i:=1 TO 16 DO
         x:=x*input/i;
         y:=y+x
      OD CYCLES=3)
  END IMPERATIVE;

END.
```

Figure 3. Example DSL program

3. SEMANTICS OF DSL: THE INTERNAL FORM

The semantics of DSL can be partially defined by the internal form generated by the compiler. This form consists essentially of 3 graphs that share the same vertices. The vertices in the graphs are the operations in the DSL program.

Let P be a DSL program, V the set of operations that appear (textually) in P and W the set of variables, constants and imperative procedures in P plus some auxiliary variables. Then the internal form S of P is given by

$$S=(G_1,G_2,G_3)$$

$$G_1=(V,E_1)$$

$G_2 = (V\ W, E_2)$

$G_3 = (V, E_3)$

G_1, G_2 and G_3 are directed graphs. The edges E represent

Sequence:
e=(v,v') in E_1 <=> v before v' and v, v' in the procedural part

Data-flow:
e=(v,w) in E_2 <=> w in W is an output of v in V
e=(w,v) in E_2 <=> w in W is an input of v in V

Timing:
e=(v,v') in E_3 <=> there is a timing constraint for a group of operations
1. starting with v and ending with v'
2. ending with v and starting with v'

The vertices and edges have the following attributes:

v in V	operation(v),	e.g. ADD, AND, etc.
	type(v),	e.g. Loop beginning, IF, etc.
	condition(v),	of the form variable=value
w in W	type(w),	e.g. two_complement, fixed
	width(w),	e.g. (15..0)
	array_dim(w),	e.g. [8..1]
e in G_1	condition(e),	of the form variable=value
e in G_2	array_index(e),	the index range if w in e=(v,w) or e=(w,v) is an array, e.g. [2..1]
	width_selection(e),	selects the used bits of w in e=(v,w) or e=(w,v), e.g. (8..1)
e in G_3	timing(e),	i.e. time or cycle delay

The nodes in V (operations) are identified by a unique index, i.e. a natural number. The nodes in W (variables) are identified

by the variable name, the constant name, the procedure name or, in the case of auxiliary variables, by a name of the form SYMB_nmb (nmb a natural number). Notice that for the applicative part, G_1 does not exist, i.e. for a pure applicative specification $S=(G_2,G_3)$.

Besides the internal form S the compiler generates a symbol table and identifies the context of the variables, i.e. in which imperative or applicative parts a variable is used. Also the global constraints are included in the output.

The output of the compiler for the example of fig. 3 is given partially in fig. 4. The first vertex represents the comparison of i with 16. The index is the internal identification, the tag indicates the attribute type of the operation, in this case a 'For Test-Single' which means the single testing operation of a FOR loop. Operation GRT stands for 'GReater Than'. The inputs are the variable i of type LOGICAL and the constant 16; the output is the internal generated variable SYMB_2 of undefined width. Both inputs and outputs define G_2. Predecessor and successor give the edges of G_1, i.e. the index of the corresponding vertices. The succesor edges have the attribute condition, in case of the operation GRT the value of the output. Finally, the timing constraint graph G_3 is given indicating the edge (index of the vertex) and the attribute (EQuaL 3 cycles).

The second operation is the MULtiplication of x and input, its type is 'Expression Begin' and it has no local timing constraints. The complete internal form is given in figure 5 in a graphical representation, with attributes attached to the vertices and edges, and the identification of the nodes inside them. For operations these are natural numbers, for variables their names.

```
Index       : 8       Tag : FTS
Operation   : GRT
Inputs      : i(4..0) : LOG    16
Outputs     : SYMB_2(*..*) : AUX
Predecessor: 11  12
Condition   : 0      Successor : 13
Condition   : 1      Successor : *
Succ.cycle  : 3       Index : 12       Relation : EQL

Index       : 13       Tag : EB
Operation   : MUL
Inputs      : x(15..0) : FIX     input(15..0) : IN
Outputs     : SYMB_1(*..*) : AUX
Predecessor: 8
```

Figure 4. Two vertices in the compiler output for exponentiation

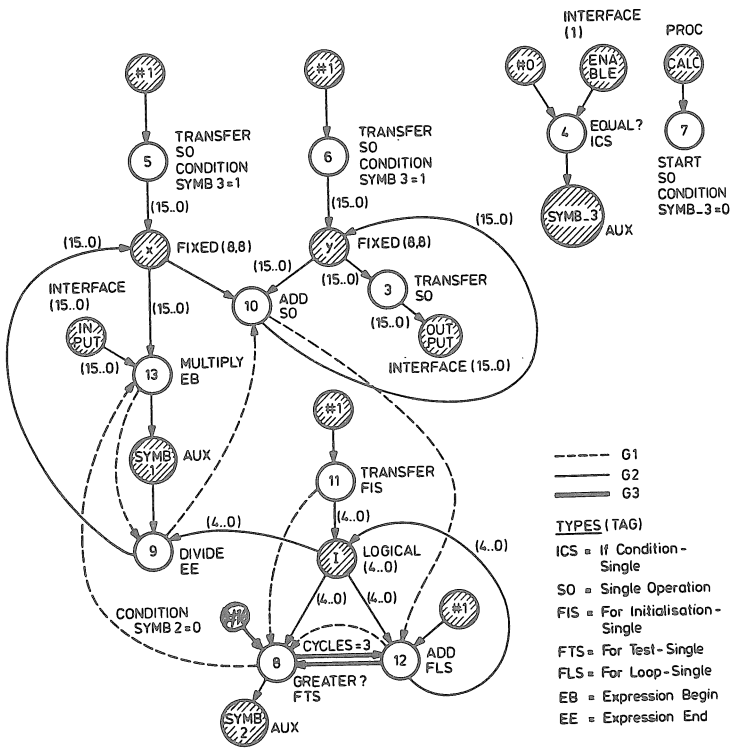


Figure 5. Internal form for exponentiation

The internal form defines an exact predecessor-successor relationship (G_1), the context of each operation (G_2) and the timing constraints (G_3). Global constraints are also included. The output of the compiler is used in subsequent steps for the synthesis of structures and for test pattern generation at the functional level. It is interesting to notice that G_2 already gives a first direct implementation of the data-path by replacing the vertices W by registers and the vertices V by the corresponding operations. G_1 provides the basis for a straight forward control part allocation by activating the operations in the sequence G_1 indicates. Of course the timing constraints given by G_3 and the global constraints must be met in subsequent optimization steps.

The semantics of DSL deliberately do not define the resulting type of many operations, e.g. what should be the type of an (intermediate) result in hardware of the division of a fixed point register by a floating point register? This kind of indefinition must be resolved at a later stage in the synthesis process, either by defining strict rules or, what seems more adequate for hardware synthesis, by interaction with the designer. Another indefinition is the result of the use of common variables in concurrent imperative processes. Also in this case many intended cases may exist, and imposing for example one synchronisation mechanism would be too restrictive. An example might be the concurrent incrementing and decrementing of a register by two processes. In case of doing this in the same cycle, the intended result might not alter it. Of course this could be specified explicitly by merging both processes, or by defining a new process that controls the access to the register, but this tends to produce clumsy specifications. Nevertheless, these situations are identified so that they can be resolved at a later time.

4. THE COMPILER

The transformation of DSL into the internal form is done by compilation. Beyond the usual tasks of a compiler, the DSL compiler achieves a certain kind of consistency of the DSL

specification by checking consistency conditions. This is very important to discover specification errors at the beginning of the design process. The cost of an error increases very fast as the design advances. The conditions implemented in this compiler include:

- Information can not be read from output interfaces or written into input interfaces
- Inside an imperative part no concurrent write or read-write is allowed
- Undefined resulting data types from intermediate operations are marked
- Variables that might generate synchronisation problems because they are used in two or more concurrent imperative parts or in the applicative part are identified
- An imperative part can not control itself

The current version of the compiler was implemented using the compiler-compiler system GAG [KHZ82]. The input to GAG is an attributed grammar in ALADIN. The output is a Pascal program, the complete compiler. Attention was paid to mantain efficiency, even though using such a high level tool. The implementation time was roughly 6 man months. The main characteristics of the compiler are

- 6400 ALADIN code lines for the source
- 27800 Pascal code lines for the generated output
- 467 KB for the object code on a Siemens 7751

The performance for some examples, including the exponentiation given in chapter 2, is shown in figure 6.

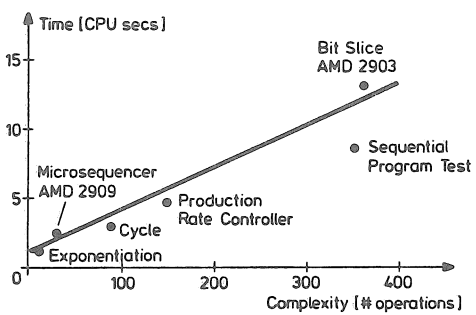


Figure 6. Time performance of the DSL compiler

5. SUMMARY

The language DSL, its internal form S and the DSL compiler were presented. The distinctive features of DSL are both applicative and imperative specification styles, modularity and hierarchies by an abstraction mechanism for functions, specification of concurrency at the operation and at the process level and the inclusion of timing constraints and global electrical constraints. The internal form consists mainly in 3 graphs with common vertices (the operations) which represent the predecessor-successor relationship, the data path and the local timing constraints. A relatively efficient version of the compiler was implemented. Consistency conditions permit the discovery of some specification errors.

The main applications of DSL are the synthesis of structures from behavioural level specifications and the test pattern generation at a functional level. For both applications partial results have already been reported and are currently under development. Interesting problems among others include the properties operations must have to allow a test pattern generation at this level, which synthesis techniques to use to obtain results that meet the specified constraints and how to integrate test and synthesis.

6. REFERENCES

[BBCS79] M.R. Barbacci, G.E. Barnes, R.G. Cattell, D.P. Siewiorek
The ISPS Computer Description Language
Department of Computer Science,
Carnegie-Mellon University, CMU-CS-79-137, 1979

[CaRo80] R. Camposano, W. Rosenstiel
Algorithmische Synthese deterministischer (Petri-) Netze aus Ablaufbeschreibungen digitaler Systeme
Interner Bericht Nr. 22/80, Fakultaet fuer Informatik, Universitaet Karlsruhe, 1980

[CaTr84] R. Camposano, L. Treff
STRUDEL: Eine Sprache zur Spezifikation
der Struktur digitaler Schaltungen
Interner Bericht Nr. 7/84, Fakultaet fuer
Informatik, Universitaet Karlsruhe, 1984

[CaWe84] R. Camposano, R. Weber
DSL - Eine Sprache zur Spezifikation
digitaler Schaltungen
Interner Bericht Nr. 24/84, Fakultaet fuer
Informatik, Universitaet Karlsruhe, 1984

[CaWe85] R. Camposano, R. Weber
Semantik und interne Form von DSL
Interner Bericht 3/85, Fakultaet fuer
Informatik, Universitaet Karlsruhe, 1985

[CKR84a] R. Camposano, A. Kunzmann, W. Rosenstiel
Automatic Data Path Synthesis from
Behavioural Level Descriptions in DSL
Proc. VLSI: Algorithms and Architectures
Edited by P. Bertolazzi, F. Lucio
North Holland, 1984

[CKR84b] R. Camposano, A. Kunzmann, W. Rosenstiel
Automatic Data Path Synthesis from DSL
Specifications
Int. Conference on Computer Design ICCD'84
Port Chester, October 1984

[DuDi68] J.R. Duley, D.L. Dietmeyer
A Digital System Design Language (DDL)
IEEE Transactions on Computers,
Volume C-17, September 1968

[DPST81] S.W. Director, A.C. Parker, D.P. Siewiorek,
D.E. Thomas
A Design Methodology and Computer Aids for
Digital VLSI Systems
IEEE Transactions on Circuits and Systems,
Volume CAS-28, Number 7, July 1981

[GiKn84] Girczyc,E.F., Knight,R.P.
An Ada To Standard Cell Hardware Compiler
based on Graph Grammars and Scheduling
Proc. Internatinal Conf. Computer Design 1984
Port Chester, N.Y., October 1984

[GHHS84] Göttler,E., Haschigh,L., Hörbst,E.,
Sandweg,G, et al
Entwicklung von kundenspezifischen
Schaltungen
Elektronik, Hefte 19,20,21,22, 1984

[John83] Johnson, Steven D.
Synthesis of Digital Designs
from Recursion Equations
MIT Press, 1984

[KoTh83b] T.J. Kowalsky, D.E. Thomas
The VLSI Design Automation Assistant:
Prototype System
20th Design Automation Conference, IEEE,
August 1983

[KuWu84] Kunzmann,Arno, Wunderlich,Hans J.
Steigerung der Effizienz beim Test mit
Zufallsmustern
Report 19/84 of the Faculty for Informatics,
University of Karlsruhe, October 1984

[KHZ82] U. Kastens, B. Hutt, E. Zimmermann
GAG: A Practical Compiler Generator
Lecture Notes in Computer Science 141
Springer Verlag, 1982

[Lieb84b] Lieberherr, Karl J.
Chip Design in Zeus and Proposal for
Standard Benchmark Set for HDL
Proc. International Conf. on Computer Design
ICCD'84, Port Chester, N.Y., October 1984

[Marw84] Marwedel, Peter
The MIMOLA Design System:
Tools for the Design of Digital Processors
Proc. of the 21st Design Automation Conf.,
Albuquerque, New Mexico, June 1984

[MeCo80] C. Mead, L. Conway
Introduction to VLSI Systems
Addison-Wesley, 1980

[Nash84] Nash, J.D.
Bibliography of Hardware Description
Languages
SIGDA Newsletter, Vol. 14, February 1984

[Rose82] W. Rosenstiel
DSL - Eine Sprache zur Spezifikation der Funktion digitaler Systeme : Konzept und Implementierung
Interner Bericht Nr. 9/82, Fakultaet fuer Informatik, Universitaet Karlsruhe, 1982

[Rose84] Rosenstiel, Wolfgang
Synthese des Datenflusses digitaler Schaltungen aus formalen Spezifikationen
Dissertation at the Faculty for Informatics, University of Karlsruhe, VDI-Verlag, 1984

[Shiv83] S.G. Shiva
Automatic Hardware Synthesis
Proceedings of the IEEE,
Volume 71, Number 1, January 1983

[Waxm84] Waxman, R.
The Many Languages of Electronic Computer Aided Design
Proc. International Conf. on Computer Design ICCD'84, Port Chester, N.Y., October 1984

[Wund84] Wunderlich, Hans J.
Zur statistischen Analyse der Testbarkeit digitaler Schaltungen
Report 18/84 at the Faculty for Informatics, University of Karlsruhe, August 1984

ESTRATÉGIAS BÁSICAS PARA A CONCEPÇÃO AUTOMÁTICA DO LAYOUT DE CIRCUITOS EM LÓGICA ALEATÓRIA

R. REIS*

SUMARIO

As estratégias básicas para a concepção automática do layout (desenho) de blocos funcionais em lógica aleatória são baseadas em três aspectos: maleabilidade, estrutura em bandas e transparência das células. É feita uma análise sobre cada um destes aspectos, apresentando como resultado algumas estratégias de concepção.

ABSTRACT

The basic strategies for the design automation of the lay-out of random logic blocks are based in three aspects: maleability, strip structure and cell transparency. It is done an analysis about each one of these aspects and some design strategies are shown as results.

* Engenheiro Eletrônico (UFRGS, 1978), Pós-Graduação em Ciência da Computação (UFRGS, 1979), D.E.A. em Microeletrônica (INPG, FRANÇA, 1980), Docteur-Ingénieur em Informática, opção Microeletrônica (INPG, França, 1983).

Áreas de interesse: Concepção de circuitos integrados, compiladores de silício, CAD;

Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Caixa Postal 1501 - 90000 - Porto Alegre - RS - Brasil.

I. INTRODUÇÃO

A evolução do processo de concepção de circuitos em alta escala de integração (VLSI) mostra que a tendência atual é o uso intenso de estruturas regulares. É muito menos complexo o desenvolvimento de ferramentas de PAC para estruturas regulares do que para estruturas em lógica aleatória. Devido ao grande número de informações que devem ser tratadas, a concepção de circuitos VLSI requer uma automação do processo de concepção, sendo por isso que as estruturas regulares são as mais utilizadas atualmente.

O uso de lógica aleatória é interessante na concepção de partes de controle (sequenciadores), pois estruturas em lógica aleatória ocupam menos área do que estruturas regulares equivalentes (PLA, ROM) /REI 83a/.

Atualmente não existem ferramentas de PAC para estruturas em lógica aleatória que sejam satisfatórias, pois a área resultante é muito maior do que no desenho manual. Isto ocorre pelo seguinte fato: o número de regras necessárias para a automação da concepção de estruturas regulares não é muito grande, pois a estrutura básica é sempre a mesma, por outro lado o número de regras relativas à concepção de blocos em lógica é muito grande, pois o lay-out pode ser efetuado de inúmeras maneiras.

Por exemplo, o layout de uma célula elementar como um inversor é flexível e quanto mais complexa é a célula mais flexível se torna o seu desenho. Fato este que torna mais complexo o processo de determinação de regras de automação. O conceptor ao fazer o desenho (lay-out) de uma célula ele faz uma série de opções (posicionamento de transistores, conexões, etc...) de maneira empírica, e de acordo com sua experiência. O computador no entanto necessita de regras objetivas e formais. Uma maneira de facilitar o processo de automação é a determinação de estratégias, denominadas de metaregras /REI 84B/

Portanto, ferramentas eficientes para a concepção automática de blocos em lógica aleatória deverão ser desenvolvidas utilizando técnicas de inteligência artificial. Um passo importante neste sentido é a determinação de regras e metaregras de concepção.

Este trabalho presente ser uma contribuição inicial na análise e na determinação de estratégias de concepção de blocos em lógica aleatória.

Nota: todos os dados citados neste trabalho são referentes à tecnologia NMOS, salvo citação em contrário.

II. MALEABILIDADE

A metodologia de concepção apresentada em /REI 83A, /REI 83F/ e /REI 84A/ é uma metodologia descendente tendo como característica o planejamento da planta baixa. Este planejamento nos permite determinar a área e a forma de cada bloco funcional antes da concepção detalhada do lay-out de cada um desses blocos. A alocação dos blocos funcionais sobre a planta baixa é feita de maneira que os blocos menos maleáveis, quanto a forma, sejam alocados prioritariamente. A maleabilidade de um bloco é definida como sendo a capacidade

de modificação de sua forma, através da modificação de sua organização interna. Um bloco funcional que constitue um circuito integrado pode ser classificado quanto ao seu grau de maleabilidade topológica, variando de "duro" a "mole". Como exemplo de blocos duros temos os "pads" de um circuito. Um bloco funcional que já esteja desenhado (já utilizado em um outro circuito) também será considerado um bloco duro. Um bloco ROM também pode ser considerado um bloco com grande grau de dureza, pois a variação de sua forma é fortemente descontínua, sendo função da variação da organização interna. Por exemplo, um plano de ROM de 4 x 4 bits pode ser modificado para 2 x 8, 8 x 2, 1 x 16 ou 16 x 1, não aceitando soluções intermediárias. Em /REI 83a/ são mostradas as possibilidades de variação da forma de diversos blocos funcionais, com a apresentação de exemplos.

O limite de maleabilidade de um bloco é um aspecto importante a ser considerado. Definimos como limite de maleabilidade a situação em que o aumento de uma das dimensões de um bloco ou de uma célula não permite reduzir a dimensão oposta. Por exemplo, se considerarmos uma célula registro à 6 transistores, podemos obter diversas soluções para o seu desenho. Em /REI 83f/ são mostradas as soluções encontradas e na figura 1, podemos observar a curva de deformabilidade da célula. Os pontos correspondentes às soluções E e C representam os limites de maleabilidade da célula em altura e largura, respectivamente. A solução A representa a célula de menor área, sendo que a curva tracejada representa uma equisuperfície.

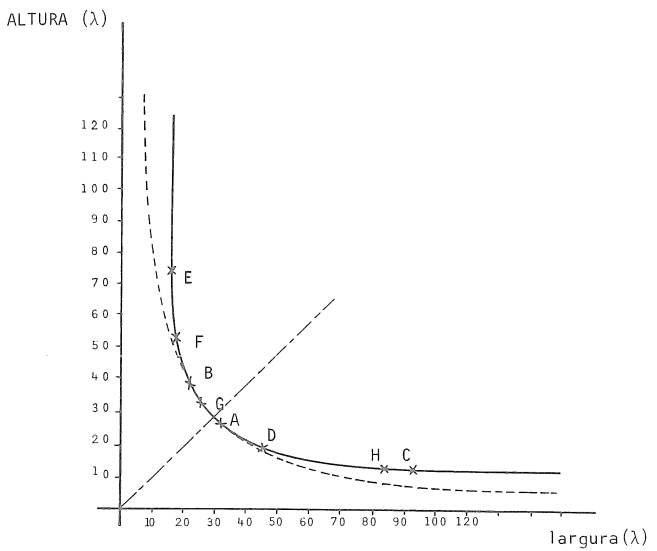


Figura 1 - Maleabilidade de um ponto de registro à 6 transistores.

Os blocos funcionais implementados em lógica aleatória são blocos que apresentam uma grande flexibilidade quanto as suas formas, sendo portanto classificados como blocos moles (caso típico). Devido a isto, devem ser os últimos blocos a serem alocados na planta baixa, procurando-se preencher os espaços vazios existentes. É interessante muitas vezes subdividir um bloco em diversas partes para melhor se adaptar à planta baixa. A forma de um bloco em lógica aleatória é, consequentemente, diretamente dependente da topologia dos outros blocos. Portanto podemos dizer que os blocos "duros" impõem restrições aos blocos "moles".

III. ESTRUTURAS EM BANDAS

Uma banda é definida como a região delimitada por duas linhas paralelas de alimentação em metal: uma linha de VCC e uma linha de massa (figura 2). Entre as duas linhas de alimentação são dispostas linhas internas em metal que servem como conexão locais (em um bloco ou célula) ou como conexões externas. A distância entre as linhas de metal é determinada pelo passo de metal (que varia em função da tecnologia) /REI 83b/.

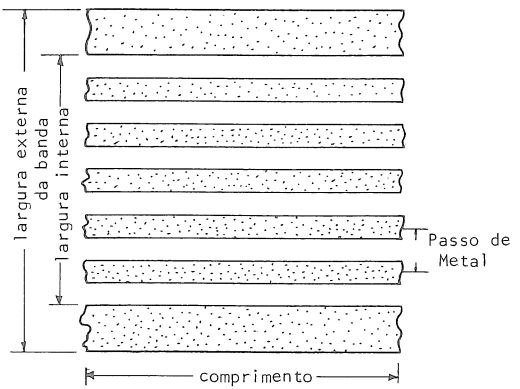


Figura 2 - Estrutura em Bandas

Para obtermos regras e metaregras de concepção devemos analisar o lay-out de circuito implementados (comerciais ou protótipos). A observação de diversos destes circuitos /REI 83a/ nos mostrou que os blocos em lógica aleatória apresentam estruturas internas sob a forma de bandas. Em alguns, o número de linhas internas varia de banda para banda, enquanto que em outros circuitos o número de linhas internas é praticamente constante, como no caso do microprocessador MC 6809 (figura 3).

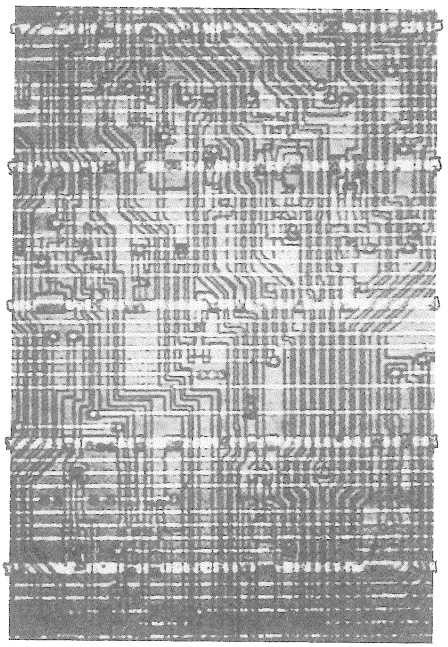


Figura 3 - Foto de um conjunto de bandas do MC6809

Um problema a ser resolvido é quanto a determinação do número ideal de linhas internas de cada banda na concepção de um determinado circuito. Uma estatística sobre o número de linhas internas por banda, de blocos em lógica aleatória de microprocessadores reais, constitue um dado importante nesta procura. Um estudo estatístico detalhado é apresentado em /REI 83a/ e um resumo é apresentado na figura 4, onde podemos observar a variação do número de linhas internas (NL) por banda em seis microprocessadores diferentes. Apesar de, na maioria dos circuitos, observarmos uma variação bastante grande do número de linhas internas, verificamos também que, em média, mais de 50% das bandas existentes em um circuito apresentam apenas dois números diferentes de linhas internas. O número médio de linhas por banda, nos circuitos observados, é 7. Devemos observar também que o número de linhas por banda aumenta com o aumento da complexidade (em nº de transistores) do bloco funcional ou do circuito.

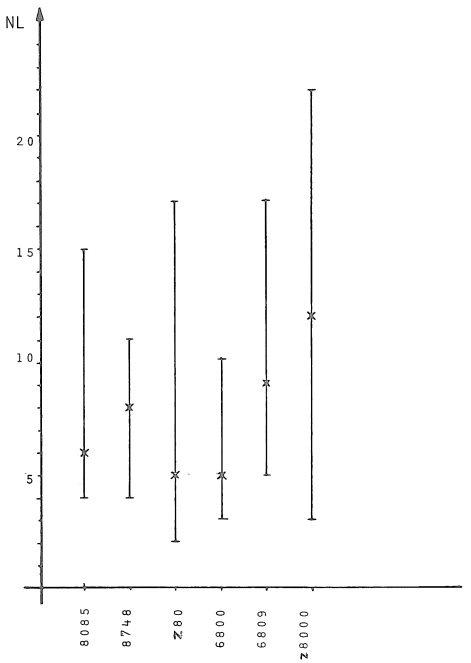


Figura 4 - Amplitude de variação do número de linhas internas (NL) por banda. A cruz indica o número médio de NL para cada microprocessador.

IV. TRANSPARÊNCIA

A transparência de uma célula ou de um bloco, em uma direção, é definida como sendo a relação entre o número de conexões externas à esta célula (ou bloco), que podem atravessá-la nesta direção, e o número de linhas internas que podem existir na célula (ou bloco) na direção considerada.

Nota: Em nossa análise chamaremos de transparência horizontal a transparência na direção passo de metal, ou seja transparência relativa às linhas horizontais (metálicas). A transparência vertical é portanto a transparência na direção passo de poli (passo de silício policristalino), ou seja, transparência relativa as linhas verticais (poli ou difusão).

Se considerarmos o exemplo de uma célula NAND (NÃO-E), que pode ser desenhada de diversas maneiras, veremos que é possível obter diversos níveis de transparência para uma mesma célula. Na figura 5, a célula NAND foi desenhada de quatro maneiras, conservando sempre a mesma forma e tamanho dos transistores. Variando o material das entradas e saídas da célula, obtemos diversos graus de transparência.

A transparência vertical de uma célula é normalmente baixa e bastante inferior à transparência horizontal. No exemplo da figura 5 temos uma transparência vertical nula em todos os casos.

Quando a transparência em uma direção é baixa, a maioria das interconexões devem utilizar canais de conexão externos. Por outro lado, quando a transparência horizontal é forte, a maioria das interconexões podem se dispor sobre as bandas, ou seja sobre os circuitos ativos (transistores).

Como a transparência horizontal é mais importante, principalmente devido ao fato das linhas metálicas serem utilizadas exclusivamente para conexões (incluindo as linhas de alimentação), nos deteremos aqui na análise da transparência horizontal. Em /REI 83a/ são apresentados alguns estudos sobre a transparência vertical.

IV.1 Transparência Horizontal

As linhas metálicas constituem a rede de conexões mais importante em um circuito. Estas conexões podem ser de diferentes classes:

- conexões internas (locais) à célula,
- conexões entre células vizinhas,
- conexões internas à um grupo de células,
- conexões externas à um grupo de células,
- ligações entre conexões realizadas em silício policristalino ou em difusão.

A transparência horizontal também pode ser dividida em diversas classes:

- transparência de uma célula,
- transparência de um grupo de células,
- transparência global de uma banda.

A primeira classe é analisada através de um estudo estatístico /REI 83a/ sobre a transparência média das células dos microprocessadores Z8000 e 6800.

A segunda e a terceira classe são analisadas através de um estudo sobre a probabilidade de utilização das linhas internas de uma banda em função do seu comprimento (medido em número de passos de poli).

IV.1.1 Transparência média de uma Célula

Esta estatística foi efetuada da seguinte maneira: primeiramente foram escolhidos, aleatoriamente, diversos grupos de células sobre bandas diferentes, com número de linhas

TRANSPARÊNCIA
100 %

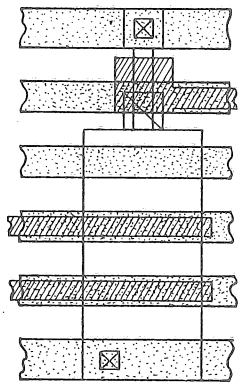


FIG. 5A

TRANSPARÊNCIA:
50 %

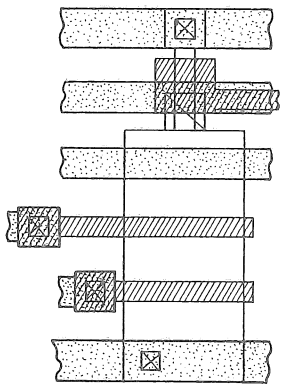


FIG. 5B

TRANSPARÊNCIA:
25 %

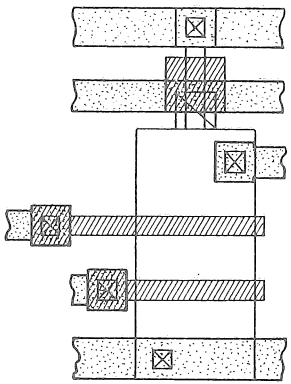


FIG. 5C

TRANSPARÊNCIA:
0 %

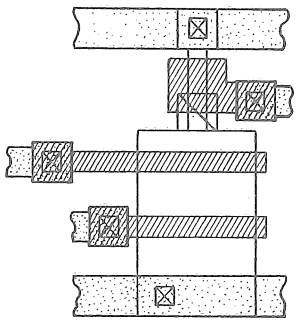


FIG. 5D

Figura 5: Celula NAND com diferentes transparências horizontais

internas (NL) diferentes. Em seguida foi medida a transparência de cada célula e obtida a transparência média de cada grupo de células com um mesmo número de linhas internas. Considerando o número de linhas internas por banda, obteve-se também uma transparência média global das células observadas. O resumo dos resultados obtidos é mostrado na tabela 1, para o microprocessador Z8000, e na tabela 2, para o microprocessador MC 6800.

NL	NÚMERO MÉDIO DE CANAIS OCUPADOS	TRANSPARÊNCIA %
12	3,64	70
9	2,86	68
7	2,41	66
5	1,67	67
Média 9,36	2,98	68

Tabela 1 - Transparência horizontal das células do Z8000.

NL	NÚMERO MÉDIO DE CANAIS OCUPADOS	TRANSPARÊNCIA %
10	3,71	63
7	1,63	77
5	1,59	68
Média 7,34	2,32	68

Tabela 2 - Transparência horizontal de células do MC 6800.

Podemos observar que mesmo variando o número de linhas internas da banda, encontramos transparências próximas de 2/3 do número de linhas. Verificamos que tanto para o Z8000 (16 bits) como para o MC6800 (8 bits) encontramos resultantes semelhantes. Observações rápidas em outros microcircuitos nos mostraram resultados semelhantes, o que nos leva a considerar que, para circuitos com complexidade acima de mil transistores, a utilização de células com 70% de transparência é uma boa solução.

IV.1.2 Probabilidade de utilização das linhas de uma banda

O estudo estatístico efetuado nos mostra o número necessário de células em uma banda para obtermos uma ocupação total de todas as linhas internas desta banda, ou seja, obter uma transparência nula.

Os resultados do estudo estatístico efetuado sobre as bandas do Z8000 /REI 83a/ são mostrados na figura 6, onde podemos observar as curvas de transparência em função do comprimento da banda (em número de passos de poli), para diferentes número de linhas internas por banda. A transparência sobre a curva é dada em número de linhas disponíveis. A lei de probabilidade obtida é do tipo exponencial, sendo que a distribuição é dada pela seguinte equação:

$$T = NL \cdot e^{-\frac{L}{20}}$$

onde:

T = transparência em número de canais livres.
NL = número de linhas internas da banda
L = comprimento da banda em número de passos de poli.

O estudo efetuado sobre o microprocessador MC 6800 apresenta resultados semelhantes (figura 7) /REI 83a/, sendo que as curvas obtidas correspondem à equação acima.

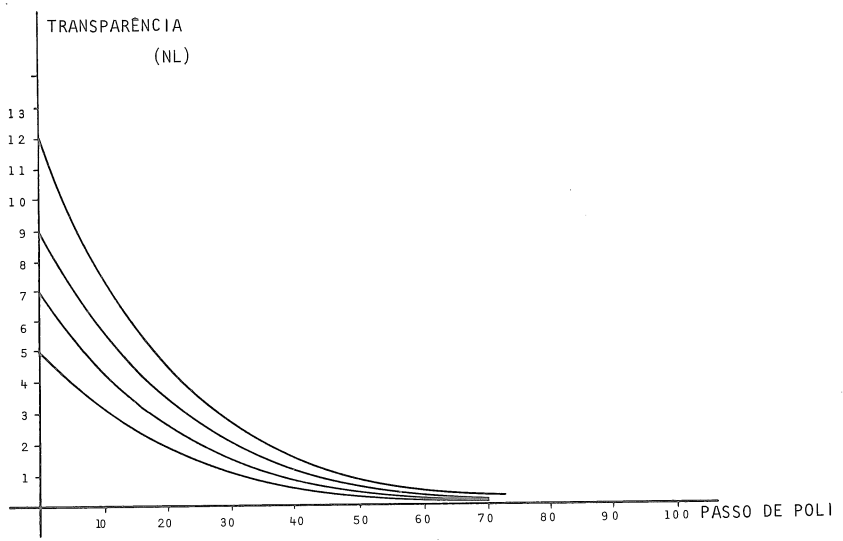


Figura 6 - Probabilidade de Transparência em função do comprimento da banda (Z8000).

V. ESTRATÉGIAS BÁSICAS

A estratégia fundamental consiste no posicionamento do maior número possível de conexões sobre as zonas ativas, (transistores) utilizando a camada em metal e a transparência horizontal das células /REI 82c/ e /REI 83c/.

O primeiro passo no desenho de um bloco em lógica aleatória é determinar a estrutura da camada em metal, ou seja, determinar a estrutura em bandas adequada. Para tanto é necessário uma definição quanto ao número de bandas e quanto ao número de canais (linhas internas) por banda. É necessário também um gerenciamento destes canais, ou seja, organizar a alocação dos mesmos para conexões locais ou externas. Este assunto é abordado no item seguinte.

Na etapa seguinte deve ser efetuado o posicionamento das células sobre a estrutura em bandas definida anteriormente. Para tanto é necessário efetuar uma avaliação da area de cada célula à partir do esquema lógico da mesma. De posse destas avaliações procede-se a "montagem" das células do bloco funcional, ou seja, reservar o espaço correspondente à cada uma das células.

O próximo passo é realizar o desenho (lay-out) de cada célula, obedecendo à seguinte ordem geral (estudo resumido relativo a tecnologia NMOS):

1 - posicionamento do transistor de carga de uma porta lógica.
2 - posicionamento dos transistores de sinal da porta lógica.
3 - posicionamento da porta lógica seguinte conforme as etapas 1 e 2.
4 - traçar as ligações em polisilício e difusão entre as portas lógicas, caso existam.
5 - Idem, porém relativo as ligações em metal.
6 - repetir os itens 3, 4 e 5 até completar a célula.

Este procedimento é repetido no desenho da célula seguinte. No próximo passo deverão ser efetuadas as ligações entre células (poli, difusão e metal). É assim por diante até completar o desenho do bloco funcional.

Em todas etapas do desenho das células e das conexões entre elas deverão ser constantemente consultados os bancos de dados que armazenam as regras de desenho e as regras elétricas. No desenho dos transistores poderá ser adotado dois processos diferentes:

A - Desenho dos transistores à partir das regras de desenho
B - Desenho dos transistores à partir de padrões elementares catalogados em uma biblioteca.

VI. GERENCIAMENTO DOS CANAIS DE UMA BANDA

Um dos objetivos deste gerenciamento é evitar ao máximo os casos em que uma conexão externa que atravessa uma célula em um canal (utilizando uma linha interna da banda) deve trocar de canal para passar sobre a célula seguinte devido a uma conexão local, o que

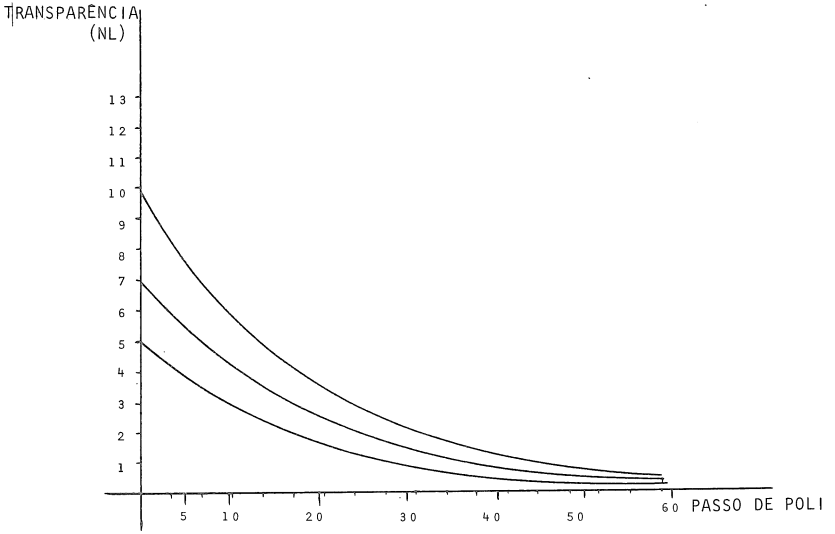


Figura 7 - Probabilidade de transparência em função do comprimento da banda (MC 6800)

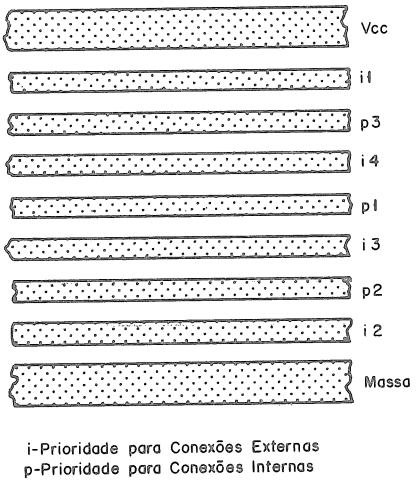


Figura 8 - Exemplo de uma Ordem de Prioridades em uma Banda.

implica na realização de uma linha vertical (em polisicio ou difusão) para ligar as duas linhas metálicas. Implica, portanto, na perda de área de silício e muitas vezes inviabiliza a passagem de uma conexão sobre um grupo de células.

O gerenciamento dos canais é baseado em um esquema de prioridades que define quais os canais tem prioridade para conexões internas e quais tem prioridade para conexões externas.

Na figura 8 apresentamos um exemplo de uma banda com 7 linhas (canais) onde as linhas impares (i) tem prioridade para conexões externas e as linhas pares (p) tem prioridade para conexões internas às células. Como as conexões internas são as primeiras a serem implementadas, caso ocupem todos os canais prioritários a conexões internas, poderão ocupar os canais com prioridade para conexões externas, procurando respeitar a ordem de prioridade (i4, i3, ...). (neste caso, i4 seria a primeira a ser ocupada). Portanto, p1 é o canal com maior probabilidade de ser ocupado por uma conexão interna e il é o canal com maior probabilidade de ser ocupado por uma conexão externa.

VII. MULTICAMADAS DE INTERCONEXÕES

As tecnologias de fabricação de circuitos integrados digitais MOS tem apresentado um desenvolvimento significativo quanto à realização de diversas camadas de interconexões, seja em silício policristalino, em metal, assim como em novos elementos físicos.

A existência destes multiníveis de conexões aumenta extraordinariamente a transparência das células. Podemos então reservar as camadas de conexões inferiores para as conexões locais de uma célula e para as conexões entre células vizinhas. Com isto, a transparência da camada inferior poderá ser nula e procurar-se-á desenhar as células de maneira que ocupem a menor área possível.

VIII. CONCLUSÃO

O desenvolvimento de um sistema capaz de gerar automaticamente o desenho de um bloco funcional em lógica aleatória, de maneira que a área resultante seja próxima à área ocupada no desenho manual, poderá inverter a tendência atual quanto ao uso preferencial de blocos funcionais com estruturas regulares (ROM, PLA, etc...).

O nosso objetivo é de levar adiante o trabalho aqui apresentado, no sentido de desenvolver uma ferramenta de PAC que realize a geração automática de blocos em lógica aleatória.

IX. BIBLIOGRAFIA

/REI 82a/ REIS, R.A.L. A Topological Evaluator as the first step in VLSI design. Proceedings of Microelectronics'82. Adelaide, South Austrália, 12-14 de maio de 1982.

/REI 82b/ REIS, R.A.L. Tess: A Topological Evaluator tool. Proceedings of ICCC 82, IEEE International Conference on Circuits and Computers. Nova York, E.U.A. 28 de setembro/1º de outubro de 1982.

/REI 82c/ REIS, R.A.L. & ANCEAU, F. Complex Integrated Circuit Design Strategy. IEEE Journal of Solid State Circuits, Vol.SC-17, nº 3. pp.459/464, junho de 1982.

/REI 83a/ REIS, R.A.L. Tess: Evaluateur Topologique Predictif pour la Generation Automatique des Plans de Masse de Circuits VLSI. Thèse de Docteur-Ingénieur em Informática. Opção Microeletrônica, INPG-Grenoble-França, 11 de janeiro de 1983.

/REI 83b/ REIS, R.A.L. Tess: A Topological Evaluator Tool for VLSI Circuits. Proceedings of 1983 International Symposium on VLSI Technology, Systems and Applications. Taipei, Taiwan, 30 de março a 1º de abril de 1983.

/REI 83c/ REIS, R.A.L. & ANCEAU, F. Design Strategy for VLSI. Livro "VLSI Architecture", Editores: B. RANDELL e P.C. TRELEAVEN. Prentice-Hall, 1983.

/REI 83d/ REIS, R.A.L. Tess: Uma Ferramenta para a Avaliação Topológica de circuitos VLSI. Anais do III Simpósio Brasileiro de Microeletrônica, USP, São Paulo, 25 a 17 de julho de 1983.

/REI 83e/ REIS. R.A.L. Microeletrônica, Nova Revolução na Economia Mundial: Continuaremos Dependentes? Anais do XVI Congresso Nacional de Informática, SUCESU, São Paulo, 17 a 23 de outubro de 1983.

/REI 83f/ REIS, R.A.L. Uma Metodologia Descendente de Concepção de Circuitos Integrados. Anais do I Simpósio Brasileiro de Concepção de Circuitos Integrados SBCCI 83. Porto Alegre, 7 a 11 de novembro de 1983.

/REI 83g/ REIS. R.A.L. Avaliação Topológica de Circuitos VLSI (trabalho convidado). II Jornadas de Diseño Lógico. Barcelona, Espanha, 14 a 16 de dezembro de 1983.

/REI 84a/ REIS, R.A.L; SUZIM, A.A. & ZYSMAN, E. Nova Etapa na Concepção de C.I.: Compiladores de Silício. Anais do IV Simpósio Brasileiro de Microeletrônica, USP, São Paulo, 24 a 26 de julho de 1984.

/REI 84b/ REIS, R.A.L. Inteligência Artificial e Concepção de Circuitos VLSI. Resumos dos trabalhos do I Simpósio Brasileiro de Inteligência Artificial, UFRGS, Porto Alegre, 12 a 14 de novembro de 1984.

PROJETO DE UMA PASTILHA TESTE PARA AVALIAÇÃO DE CI DIGITAIS FABRICADOS SEGUNDO A TECNOLOGIA HMOS

M.STRUM *
L.S.ZASNICOFF **

SUMÁRIO

Este trabalho descreve o projeto de uma pastilha contendo 74 estruturas de teste que servem para suporte do projeto (parâmetros) bem como para avaliação do desempenho de CI digitais fabricados segundo a tecnologia HMOS. A pastilha foi projetada baseada em normas sugeridas pelo National Bureau of Standards americana e foi prevista a extração de dados e de parâmetros por sistema automático de medidas. Esta pastilha será utilizada para avaliar o desempenho de uma memória RAM estática HMOS que está sendo desenvolvida no LME-USP, bem como uma versão otimizada da tecnologia HMOS já desenvolvida no LME que inclui um contato entre a camada de alumínio e o substrato.

ABSTRACT

This work describes a test chip design containing 74 test structures which will provide informations (parameters) for circuit design and will also be used to evaluate HMOS digital circuits performance. This chip has been designed based on the NBS suggested standards and automatic parameter extraction has been predicted. This chip will be used to evaluate an HMOS static RAM performance which in being developed at LME-USP. It will also be used to evaluate a new version of an HMOS process developed at LME/USP which includes a contact between the aluminum layer and the bulk.

* Engenheiro Eletricista, Mestre e Doutor em Engenharia (1971,1978,1983); Projeto e Fabricação de CI Digitais MOS.

** Engenheiro Eletricista, Mestre em Engenharia (1970,1974); Projeto e Fabricação de CI Digitais MOS.

Laboratório de Microeletrônica da USP - Caixa Postal 8174
01000 SÃO PAULO - SP - fone: (011) 815.93.22 ramal 310.

INTRODUÇÃO

Com o advento de circuitos de altíssima densidade, correspondendo à integração em escala muito ampla (VLSI), evidenciou-se a importância do problema de testabilidade, com um aspecto essencial no projeto de circuitos integrados . Existem dois tipos de teste que devem ser distinguidos: teste funcional de um circuito e teste de avaliação. Usamos o termo "testes de avaliação" para abranger todos aqueles que se realizam com o objetivo de se obter informações utilizadas tanto na fase de projeto do circuito (parâmetros elétricos e tecnológicos necessários à simulação lógica e elétrica do circuito e definição do lay-out), como na fase de caracterização do desempenho de circuito (influência de parâmetros e dados de rendimento) {1 a 33}. Tais informações são obtidas a partir de estruturas de teste especialmente projetadas para este fim.

Estas são arranjadas em uma pastilha denominada pastilha teste. Podem ser fabricadas em algumas posições estratégicas da mesma lâmina de silício onde estão os circuitos, caso em que servem para monitoração do processo ("process monitor chips" PMC) {8} ou, então, em lâminas de silício compostas exclusivamente destas pastilhas denominadas "lâminas de validação do processo" ("process validation wafer" PVW) {5,10}. As estruturas contidas na pastilha teste devem permitir a determinação das informações procuradas a partir de medidas elétricas. Desta forma, utilizando-se sistemas automáticos de medidas, extrai-se rápida e confiavelmente uma grande quantidade de dados {1,2,3} que devidamente processados {2,7,8} fornecem as informações.

OBJETIVO

O objetivo do presente trabalho é o de projetar uma pastilha contendo estruturas de teste capazes de fornecer todas as informações necessárias para o projeto e a avaliação de CI digitais, fabricados segundo a tecnologia HMOS desenvolvida no LME. Vale ressaltar que tal tecnologia está sendo otimizada pela inclusão de um contato entre alumínio e substrato (além dos contatos alumínio-silício poli e silício poli-substrato já existentes) e, portanto, esta pastilha servirá também para avaliar o resultado desta otimização tecnológica.

CONSIDERAÇÕES SOBRE A PASTILHA TESTE

A Tabela 1 apresenta uma lista das estruturas projetadas, indicando o número de plataformas de contato, área que ocupa e informações que fornece.

Vamos em seguida descrever as estruturas que incluímos na pastilha e que não compareceram em trabalhos anteriormente publicados {34,35}.

Foram fabricados dois diodos de mesma área (160.000μm²) e de perímetros diferentes com o objetivo de se determinar as capacitâncias lateral e de fundo de junções típicas da tecnologia. Estes parâmetros comparecem no modelo do transistor MOS do simulador SPICE G5. Também há 2 transistores meandros do mesmo perímetro (2500μm), para comparar a resistência de fonte e dreno de transistores fabricados com contato Al-Si com a dos que apresentam contato Sipoli-Si. Um terceiro transistor meandro, mas de perímetro muito maior (20000μm), foi incluído para se determinar as capacitâncias de superposição necessárias para o simulador SPICE.

Este mesmo transistor apresenta área efetiva de canal de 150.000μm², idêntica à dos capacitores de óxido fino e, assim, poderemos também comparar o rendimento do processo quanto à qualidade do óxido de porta entre estas estruturas.

Duas linhas de atraso, formadas de tiras de silício poli, uma sobre óxido de campo e outro sobre óxido fino foram introduzidas a fim de termos dados experimentais que nos permitam analisar o comportamento do fator RC distribuído de tiras condutoras muito extensas. Devido à cargas capacitivas muito grandes presentes durante as medidas elétricas, as linhas de atraso apresentam saídas através de estágios seguidores de fonte.

Finalmente, queremos destacar a inclusão das estruturas em cruz e ponte (Van der Paw) para determinar a largura de tiras e resistência de folha, estruturas para determinação da continuidade de tiras condutoras (Al e Sipoli) de continuidade de contatos (Al-Sub e Poli-Sub), de isolamento (horizontal) entre tiras de um mesmo condutor, de isolamento (vertical) entre diferentes materiais condutores e de desalinhamento entre máscaras.

Estas estruturas são essenciais para determinar parâmetros de processo, bem como fornecer dados sobre o rendimento do processamento.

Uma descrição mais detalhada sobre todas as estruturas existentes encontra-se na ref.{36}.

CONCLUSÃO

Apresentamos, neste texto, a descrição do projeto de uma pastilha teste contendo estruturas para determinar todas as informações necessárias para o projeto de CI digitais HMOS, bem como para a avaliação de seu desempenho. A pastilha contém um total de 74 estruturas.

No projeto da pastilha, adotamos diversas recomendações da NBS americana que citamos em seguida.

Usamos arranjos de plataformas de contato ("bonding pads") regulares para viabilizar a utilização de sistemas de medida automática acoplados a microprovadores automáticos, munidos de cartelas de pontas ("probe cards"). Escolhemos dois tipos de arranjos: um deles foi o arranjo em duas colunas (arranjo 2 x N {11, 12}) para estruturas que serão testadas exclusivamente através do microprovador, e o outro foi o arranjo retangular que permite também medir as estruturas encapsuladas. No primeiro caso , adotamos colunas de 15 plataformas e, no segundo, retângulos de 24 plataformas (v. fig.1).

Adotamos como dimensões mínimas para as plataformas de contato, quadrados de 80μm de lado, separados de 80μm {11}. Então, no arranjo em colunas, estas foram as dimensões usadas, e no arranjo retangular usamos separação entre plataformas de 90μm para maximizar a área útil onde estão as estruturas (v.fig.1).

Finalmente, procuramos minimizar o uso de diferentes estruturas contendo plataformas comuns {11} . Em toda a pastilha, isto só ocorre em alguns conjuntos de 5 transistores que apresentam um terminal comum (fonte ou dreno).

Considerando o número total de plataformas de solda necessárias para todas as estruturas, a área total ocupada (v.tabela 1) e devido à limitação dos equipamentos existentes no LME-USP não permitir a fabricação de pastilhas com área superior a um quadrado de 3mm de lado, distribuimos as estruturas em duas pastilhas, cada uma com área de (2950 x 3290)μm^2, conforme ilustrado na figura 1.

CONTEÚDO DA PASTILHA TESTE

A fim de definir as estruturas que constituem a pastilha teste baseamo-nos nas seguintes referências:

. a) literatura científica {1 a 33} ;
. b) documento contendo a descrição de uma pastilha teste desenvolvida no CNET - Grenoble ;
. c) experiência acumulada pelos integrantes da divisão MOS em trabalho de desenvolvimento de tecnologias e circuitos {34 e 35}.

Como resultado deste estudo, definimos as estruturas apresentadas na tabela 1. A finalidade de cada estrutura também está apresentada na mesma tabela.

Devido ao número total de plataformas de solda, da área total ocupada e à limitação dos equipamentos do LME, as estruturas ocuparam a área de 2 retângulos de 2950 x 3290μm^2 .

Esta pastilha será usada para caracterizar uma versão otimizada do processo HMOS desenvolvido no LME, bem como para avaliar o desempenho de uma memória RAM estática de 256 bits.

Nº	ESTRUTURA	CARACTERÍSTICAS	ÁREA OCUPADA	Nº PLATAFORMAS DE CONTATO
1 a 4	Capacitor MOS do óxido fino	(300x500), Substrato V_{TN}, V_{TE}, V_{TD1}, V_{TD2}	4x(300x500)μm^2	4
5	Capacitor MOS de óxido espesso	(400x600), Al/GOX+REOX + PSG/ N^+ AUTOAl.	(440x640) μm^2	2
6	Capacitor MOS de óxido espesso	(450x550), Al/REOX+PSG/POLI (500x600), POLI/FOX/V_{TF}	(500x600) μm^2	2
7	Diodo	(400x400), Al/N^+DIF/ P-campo (lateral) P- subst.(fundo), meandro	(540x770) μm^2	2
8	Diodo	(320x500), Al/N^+AUTOAL/ P-campo (lateral) P-subst. (fundo), retangular	(320x500) μm^2	1
9	Diodo	(100x100), Al/N^+AUTOAL/ P-V_{TE} (lateral) P-SUBS (fundo)	(110x100) μm^2	1
10 a 13	Trans.MOS óxi do fino e de geomet.grande	(100x100), POLI/GOX/V_{TN}, V_{TE}, V_{TD1}, V_{TD2}	4x(120x150)μm^2	4x3
14 a 43	Trans.MOS óxi do fino e de geomet.pequena	(Várias geometrias), POLI/GOX, V_{TN}, V_{TE}, V_{TD1}, V_{TD2}		6x11
44	Trans.MOS óxi do fino muito largo	(20.000x9), POLI/GOX/V_{TE}, MEANDRO	(900x1100) μm^2	3
45	Trans.MOS óxi do fino muito largo	(2500x4), POLI/GOX/V_{TE} c/CTA. AL/SUBS	(260x410) μm^2	4
46	Trans.MOS óxi do fino muito largo	(2500x4), POLI/GOX/V_{TE} c/CTA.POLI/SUBS.	(260x410) μm^2	4
47 a 50	Trans.MOS de óxido espesso	(Várias geometrias), POLI/FOX, V_{TE}		4x3
51 a 54	Inversores	2-(4x8); (8x16); (25x50)		1x4 + 3x5
55	Cascata de inversores	41 inversores c/8 saídas	(90x1070)μm^2	13
56	Oscilador em anel	55 inversores	(120x750)μm^2	4
57 e 58	Linha de transmissão	POLI/GOX e POLI/FOX c/4 saídas	2x(150x620)μm^2	2x6
59 a 63	Estr.Van der Paw e Ponte Resistiva	Largura de regiao ativa, POLI,Al R_S de POLI,AL,N+DIF,N+AUTOAL, POLI + N^+DIF		3x8 2x4
64 a 66	Estrutura de Berger	R_c de Al/SUBS , Al/POLI,POLI/SUBS.		3x6
67 e 68	Cadeia de Contatos	1000 Aberturas de Cont. {Al/SUB {POLI/SUB	2x(670x690)μm^2	1x5+1x4
69 e 70	Tiras condutoras muito longas	Al-8μm de larg. e ~6mm de compr. POLI-4μm larg. e ~6mm de compr.	2x(450x450)μm^2	2x9
71 a 74	Resistores de Desalinhamento	Região Ativa - POLI Regiao Ativa - Contato Al/SUBS. Região Ativa - Al Regiao POLI - Contato Al/SUBS.		4x10

Tabela 1a - Estruturas componentes da pastilha teste, suas principais características (dimensões e camadas constituintes), área da pastilha ocupada e número de plataformas de contato.

TABELA 1b. - Informações extraídas de cada estrutura.

INFORMAÇÕES PARA PROJETO DE CI's					INFORMAÇÕES PARA AVALIAÇÃO DE DESEMPENHO DE CI	
Número	Regras do Lay-Out	Distribuição dos parâmetros do modelo do transistor MOS no SPICE G5	Distribuição dos parâmetros de circuito	Distribuição dos parâmetros tecnológicos	Comparação entre parâmetros usados no projeto e encontrados após a fabricação do CI	Rendimento do processamento (Yield)
1 a 4		t_{ox} , N_{SUB}		$N(x), Q_{SS}, Q_O, N_{FS}, E_{FD}, t_V$	$t_{ox}, Q_{SS}, Q_O, E_{RD}$	E_{RD}, óxido perfurado
5				t_D, E_{RD}, Q_{SS}, Q_o		E_{RD}, dielétrico perfurado
6				T_{ox}		
7		C_{LJ}	V_{BR}		V_{BR}	Diodo em curto
8		C_{FJ} , J_S	V_{BR}		J_S	J_S , V_{BR}
9			V_{BR}		V_{BR}	V_{BR}
10 a 13		$V_{TN}, V_{TE}, V_{TD1}, V_{TD2}$, μ_o , $TETA_o$, GAMA $UCRIT_o, UEXP_o, UTRA_o$	V_{BR} $V_{BR}(V_{GS}), V_{PT}$		$V_{TN}, V_{TE}, V_{TD1}, V_{TD2}$ μ_o , V_{BR}, V_{PT}	V_{TN} , V_{TE}, V_{TD1} V_{TD2}, V_{BR}, V_{PT}
14 a 43	L, L_D	μ_o, UCRIT, UEXP, UTRA, V_{MAX}, F_B SIGMA, DELTA, R_S, R_D		ΔL_o , ΔL_D	μ_o , $\Delta L_o, \Delta L_D$ R_S , R_D	$R_S, R_D, \Delta L_o$, ΔL_D
44		C_{OV}				óxido perfurado
45		R_S , R_D			R_S, R_D	R_S , R_D
46		R_S , R_D			R_S, R_D	R_S , R_D
47 a 50	ΔE_{ISO}			T_{ox} , V_{TF}	V_{TF}	V_{TF}
51 a 54			$V_{TL}, \Delta O, \Delta I, t_r, t_f$			
55			t_d			F_{INV}
56			t_d, P_i, Pt			
57 e 58			RC_{par}			
59 a 63	Z_{COND}			R_F	Z_{COND}, R_F	Z_{COND}, R_F
64 a 66				R_C	R_C	R_C
67 e 68	S_{CT}				S_{CT}	S_{CT}
69 e 70	D_{COND}				D_{COND}	CN_{COND}, IS_{COND}
71 a 74	D_{FOTOM}			D_{FOTOM}	D_{FOTOM}	

LISTA DE SÍMBOLOS

C_{FJ} - capacitância de fundo da junção

C_{LJ} - capacitância lateral da junção

C_{OV} - capacitância de superposição (porta-fonte e porta dreno)

CN_{cond} - continuidade do condutor

D_{cond} - distância entre os condutores

D_{Fotom} - desalinhamento entre fotomáscaras

DELTA - fator de canal estreito

ΔE_{ISO} - mínima isolação LOCOS

E_{RO} - campo de ruptura dielétrica

F_B - fator de corpo

F_{INV} - frequência de inversores funcionando

GAMA - sensibilidade de substrato

IS_{cond} - isolação entre condutores

J_S - densidade de corrente reversa de junção

L - comprimento de canal (sem descontar a difusão lateral)

L_D - largura da difusão lateral

ΔL_O = L_M - L (L_M .... L de máscara)

ΔL_D = L - L_{ef} (L_{ef} ... L efetivo de canal)

N(x) - perfil de concentração

N_{FS} - densidade de estados rápidos de interface

N_{SUB} - concentração de substrato

P_i - potência dissipada por inversor

Pt - figura de mérito

Q_{SS} - densidade efetiva de carga de interface

Q_O - densidade de carga móvel

R_C - resistência de contato

R_D - resistência de dreno

R_F - resistência de fonte

R_S - resistência de folha (pelicular)

RC_{PAR} - parâmetros RC parasitários

S_{CT} - dimensão da abertura de contato

SIGMA - coeficiente do efeito de realimentação estática

t_d - tempo de atraso

t_f - tempo de descida

t_r - tempo de subida

t_{ox} - espessura do óxido de porta

t_V - tempo de vida dos portadores

t_D - espessura do dielétrico

T_{ox} - espessura do óxido de campo

$TETA_o$ - fator de mobilidade

$UEXP_o$ - fator de mobilidade

$UCRIT_o$ - fator de mobilidade

$UTRA_o$ - fator de mobilidade

V_{BR} - tensão de ruptura de junção

V_{PT} - tensão de perfuração entre fonte e dreno

V_{TD1} - tensão de limiar do transistor tipo depleção (só As)

V_{TD2} - tensão de limiar do transistor tipo depleção (As + B)

V_{TE} - tensão de limiar do transistor tipo enriquecimento

V_{TF} - tensão de limiar do transistor de óxido de campo

V_{TL} - tensão de limiar lógico

V_{TN} - tensão de limiar do transistor com substrato natural

V_{MAX} - velocidade máxima dos elétrons

Z_{COND} - largura de tiras condutoras

μ_o - mobilidade

Δ_o - margem de ruído nível lógico baixo

Δ_1 - margem de ruído nível lógico alto

Figura 1 - Disposição das estruturas e plataformas de contato na pastilha teste.
Desenho em escala aproximada de 1mm = 20µm

BIBLIOGRAFIA

{1} SINGER,P.H. - "Parametric Test System Update" - Semiconductor International, September 1983. pp.84-90.

{2} KAEMPF.U. - "Automated Parametric Testers to Monitor the Integrated Circuit Process" - Solid State Technology - September 1981. pp.81-87.

{3} CHRONES,C. - "Parametric Test System for Wafer Processing" - Semiconductor International, October 1980. pp.113-122.

{4} IPRI,A.C. - "Impact of Design Rule Reduction on Size, Yield, and Cost of Integrated Circuits" - Solid State Technology, February 1979, pp.85-91.

{5} RUSSELL,T.J., MAXWELL,D.B., REIMANN,C.T., BUEHLER,M.G. - "A Microelectronic Test Pattern for Measuring Uniformity of an Integrated Circuit Fabrication Technology" - Solid State Technology - February 1979. pp. 71-74.

{6} STAHLBERG,N.F. - "The Role of Testing in VLS/VHS Integrated Circuit Development" - Solid State Technology, November 1982. pp.89-95.

{7} BUEHLER,M.G., LINHOLM,L.W. - "Role of Test Chips in Coordinating Logic and Circuit Design and Layout Aids for VLSI" - Solid State Technology, September 1981, pp.68-74.

{8} PERLOFF,D.S., HASAN,T.F., BLOME.E.R. - "Real-Time Monitoring of Semiconductor Process Uniformity" - Solid State Technology - February/80. pp.81-86.

{9} BUEHLER,M.G. - "The Use of Electrical Test Structure Arrays for Integrated Circuit Process Evaluation" - Journal of the Eletrochemical Society, October 1980. pp.2284-2290.

{10} BUEHLER,M.G., SAWYER,D.E. - "Microeletronic Test Patterns" - Circuits Manufacturing - 1977, pp.46-56.

{11} BUEHLER,M.G. - "Comprehensive Test Patterns with Modular Test Structures ; The 2 by N Probe-Pad Array Approach" - Solid State Technology - October 1979. pp.89-94.

{12} BUEHLER,M.G., GRISWOLD,T.W., PINA.C.A., TIMOC,C. - "Test Chips for Custom ICs" - Circuits Manufacturing. pp.36-42.

{13} STEMP,I.J., NICHOLAS,K.H., BROCKMAN,H.E. - "Automatic Testing and Analysis of Misregistrations Found in Semiconductor Processing" - IEEE Transactions on Electron Devices - April 1979 - pp.729-732.

{14} PERLOFF,D.S. - "A Van der Pauw Resistor Structure for Determining Mask Superposition Errors on Semiconductor Slices" - Solid State Electronics, 1978. pp.1013-1018.

{15} HASAN,T.F., KATZMAN,S.U., PERLOFF,D.S. - "Automated Electrical Measurements of Registration Errors in Step-and-Repeat Optical Lithography Systems" - IEEE Transactions on Electron Devices, December 1980. pp . 2304-2312.

{16} PERLOFF,D.S. "A Four-Point Electrical Measurements Technique for Characterizing Mask Superposition Errors on Semiconductor Wafers" - IEEE Journal of Solid State Circuits, August 1978. pp.435-444.

{17} HASAN,T.F., PERLOFF,D.S. - "Automated Wafer Mapping for Characterizing Plasma Etching Processes" - Eletrochemical Society Extended Abstracts, 1979. pp.1515-1517.

{18} RUSSELL,T.J., LEEDY,T.F., MATTIS,R.L. - "A Comparison of Electrical and Visual Alignment Test Structures for Evaluating Photomask Alignment in Integrated Circuit Manufacturing" - IEDM - December 1977. pp. 7A-7F.

{19} DAVID,J.M., BUEHLER,M.G. - "A Numerical Analysis of Various Cross Sheet Resistor Test Structures" - Solid State Electronics, 1977, pp. 539-543.

{20} BUEHLER,M.G., THURBER,W.R. - "An Experimental Study of Various Cross Sheet Resistor Test Structures" - Journal of the Eletrochemical Society, April 1978. pp.645-650.

{21} BUEHLER,M.G., GRANT,S.D., THURBER,W.R. - "Bridge and Van der Pauw Sheet Resistors for Characterizing the Line Width of Conducting Layers" Journal of the Eletrochemical Society - April 1978, pp.650-654.

{22} YEN,D., LINHOLM,L.W., BUEHLER,M.G. - "A Cross-Bridge Test Structure for Evaluating the Linewidth Uniformity of an Integrated Circuit Litography System" - Journal of the Eletrochemical Society, October 1982. pp. 2313-2318.

{23} LINHOLM,L.W., BUEHLER,M.G. - "A Cross-Bridge Structure Array for Characterizing Intrachip Linewidth Variations" - Eletrochemical Society Extended Abstracts, 1979. pp.502-505.

{24} HAM,W.E. - "Intrachip and Spatial Parametric Integrity and Important part of IC Process Characterization" - IEDM - December 1977. pp.406-409.

{25} IPRI,A.C., SARACE,J.C. - "Integrated Circuit Process and Design Rule Evaluating Techniques" - RCA Review - September 1977. pp.323-351.

{26} PERLOFF,D.S., WAHL,F.E., CONRAGAN,J. - "Four-Point Sheet Resistance Measurements of Semiconductor Doping Uniformity" - Journal of the Eletrochemical Society, April 1977. pp.582-590.

{27} PERLOFF,D.S.- "Four-Point Sheet Resistance Correction Factors for Thin Rectangular Samples" - Solid State Electronics, 1977, pp.681 - 687.

{28} CARVER,G.P., BUEHLER,M.G. - "Design Considerations for the Integrated Gated-Diode Electrometer" - Eletrochemical Society Extended Abstracts, 1979. pp.508-511.

{29} CARVER,G.P., BUEHLER,M.G. - "An Analytical Expression for the Evaluation of Leakage Current in the Integrated Gated-Diode Electrometer" - IEEE Transactions on Electron Devices, December 1980. pp.2245-2252.

{30} PAUW,L.J. Van der - "A Method of Measuring Specific Resistivity and Hall Effect of discs of arbitrary shape" - Philip Res.Repts. 1958 . pp. 1 a 9.

{31} HARRIS,K., SANDLAND,P., SINGLETON,R. -"Wafer Inspection Automation: Current and Future Needs" - Solid State Technology - August 1983 . pp.199-205.

{32} PERLOFF,D.S., WAHL,F.E., MALLORY,C.L., MYLROIE,S.W. - "Microelectronic Test Chips in Integrated Circuit Manufaturing" - Solid State Technology - September 1981. pp.75-80.

{33} MALLORY,C.L., PERLOFF,D.S., HASAN,T.F. - "Spatial Yield Analysis in Integrated Circuit Manufacturing" - Solid State Technology - November 1983 - pp.121-127.

{34} ZASNICOFF,L.S. - "Desenvolvimento da tecnologia HMOS: Resultados experimentais nos dispositivos teste" - Anais do III Simpósio Brasileiro de Microeletrônica, 1983. pp.369-386.

{35} STRUM,M. - "Estudo e realização de uma célula básica de memória EEPROM" Tese de Doutoramento - USP - 1983.

{36} STRUM,M., ZASNICOFF, L.S. - Relatório Interno LME/USP - 1983.

PROJETO PARA TESTABILIDADE DE CI'S: UMA APLICAÇÃO PRÁTICA

AUTORES: R.A. Orives, A.H.C. Oliveira, F.H. Behrens, J.G.M.Taveira, A.M. Kuniyoshi

SUMÁRIO:
Este trabalho apresenta uma experiência em projeto para a testabilidade aplicada a CI's digitais complexos, contendo as técnicas utilizadas e os resultados obtidos.

Inicialmente, é feita uma descrição dos problemas encontrados no teste funcional das partes combinacional e sequencial dos CI's digitais, acompanhada da abordagem de projeto, em função de tais problemas.

Serão mostradas, então, as técnicas utilizadas no projeto, tais como:"scan test", multiplex,... etc, para a testabilidade de um circuito integrado.

Finalmente, apresentaremos os resultados obtidos, tais como o número de vetores de teste, cobertura de falhas e lógica adicional envolvida no projeto.

ABSTRACT:
This work presents an experience in design for the testability aplicated to complex digital IC's, containing the technique used and the results obtained.

First, it will be described the problems found in the functional test of the combinational and sequencial parts of digital IC's followed by the approach to the design due to such problems.

Then, it will be showed the techniques used in the design , as "scan test", multiplex, ..., etc, for the testability of an integrated circuit.

Finally, it will be presented the results obtained, as the number of test vectors, fault coverage and additional logic involved in the design .

RAMON A. ORIVES : Engenheiro Eletricista
Área de interesse: Projeto de Circuitos Integrados

ARTHUR H.C.OLIVEIRA: Engenheiro Eletricista
Área de interesse: Projeto de Circuitos Integrados

FRANK HERMAN BEHRENS: Engenheiro Eletricista
Área de interesse: Projeto de Circuitos Integrados

JOSÉ G.M. TAVEIRA: Engenheiro Eletricista
Área de interesse: Projeto de Circuitos Integrados

ANA MARIA KUNIYOSHI: Física - Mestrado em Engenharia Eletrônica
Área de Interesse: Projeto de Circuitos Integrados

ENDEREÇO PARA CORRESPONDÊNCIA: CENTRO TECNOLÓGICO PARA INFORMÁTICA - CTI
Caixa Postal 6162- Campinas, SP - CEP 13100
Fone (0192) 42-1000

INTRODUÇÃO:

O aumento do número de portas lógicas nos projetos de CI's tem gerado um acréscimo na complexidade do teste elétrico funcional; o efeito imediato deste problema é uma majoração no custo do teste final.

Algumas vezes, sacrifica-se o teste, diminuindo a cobertura de falhas em função do tempo de execução, equipamentos utilizados e custos finais. A consequência disto é a obtenção de um CI com maior probabilidade de se descobrir a falha apenas no campo, portanto, um produto com menor grau de confiabilidade.

Existem dois caminhos que podem ser seguidos para o problema teste. O primeiro caminho seria a otimização dos métodos existentes e a criação de novas técnicas de preparação de teste; este caminho implica na geração de estímulos (vetores de teste) e na configuração dos equipamentos de teste, tendo o CI com sua fase de projeto já terminada. O segundo caminho a seguir,seria projetar o CI de tal forma que sua concepção o tornasse mais facilmente testável. Este segundo caminho, contrário ao primeiro, considera o problema teste desde a concepção do CI. Pode-se chamar esta segunda abordagem de "projeto para testabilidade"; informalmente, testabilidade pode ser definida como a medida da facilidade de gerar e aplicar vetores de teste para a detecção de falhas físicas em um circuito [1].

Na maior parte dos projetos de CI's digitais complexos, o problema teste infelizmente não é abordado durante o projeto, resultando em circuitos difíceis de serem testados.

Mostra-se neste trabalho os problemas encontrados no teste funcional de CI's digitais complexos, a abordagem de projeto e as técnicas de testabilidade aplicadas a um circuito, projetado no IM-CTI, bem como os resultados obtidos.

TESTE FUNCIONAL:

O problema teste possui dois aspectos importantes: o primeiro é a geração do teste, que é o processo de se criar estímulos para exercitar e demonstrar o correto funcionamento do CI. O segundo é a verificação do teste, que é o processo da averiguação da eficiência dos estímulos gerados. [2].

A geração dos estímulos para exercitar um circuito (geração de vetores de teste) é uma tarefa que deve perseguir dois pontos essenciais que são o controle e observação dos nós internos do circuito. O controle de um nó é a capacidade de se forçar um estado lógico neste, a partir das entradas primárias (pinos) do circuito. A observação é a capacidade de se conhecer o estado lógico de um nó utilizando-se, para isso, apenas as saídas do circuito (pinos). Para se conseguir os objetivos de controle e observação é necessária a geração de vetores de teste que possibilitem a propagação dos estímulos das entradas primárias até os nós a serem controlados e também a propagação dos estados lógicos internos, os quais queremos observar, até os pinos do CI.

Circuitos digitais complexos de partes sequenciais e combinacionais tendem a possuir muitas características funcionais, tornando muitas vezes inviável o teste exaustivo da funcionalidade. Para sanar este problema lança-se mão de alguns algorítmos especialmente concebidos para este fim, podemos citar como exemplos: Algorítmo D [3], [4], Algorítmo 9V [5], Diferenças Booleanas [6] e muitos outros propostos.

É fato que a geração de vetores de testes para circuitos sequenciais tem se mostrado uma árdua tarefa para os projetistas, além disto os algorítmos existentes para circuitos puramente combinacionais na maioria dos casos têm se mostrado efetivos; o mesmo não acontecendo com os algorítmos para circuitos sequenciais. A implantação destes algorítmos em sistemas de computação permite um maior dinamismo na geração de vetores de teste; todavia, na prática, temos encontrado algumas restrições à geração automática destes. Quando se trabalha com circuitos digitais complexos, é necessário um tempo de CPU

elevado para se chegar a resultados satisfatórios. Muitas vezes, gera-se manualmente alguns vetores para se chegar à cobertura de falhas desejada. Para circuitos complexos altamente sequenciais, a geração automática se torna inviável e mesmo a geração manual se mostra extremamente difícil, geralmente resultando em baixas taxas de cobertura de falhas ou numerosos vetores de teste implicando em altos custos de teste.

PROJETO PARA TESTABILIDADE:

O projeto para testabilidade de um circuito, como foi descrito na introdução, deve começar na concepção deste e se prolongar por toda a fase de projeto.

Foi visto também que circuitos altamente sequenciais dificultam a geração de vetores de teste.

Tendo em vista estes aspectos e aliado ao fato de que a equipe de projeto do Instituto de Microeletrônica - CTI tinha pela frente um projeto de CI digital, potencialmente com alto grau de sequencialidade e obrigatoriamente uma alta taxa de cobertura de falhas, decidiu-se por uma abordagem de projeto com enfoque para a testabilidade.

O ponto chave no projeto era diminuir, se não eliminar, a sequencialidade do circuito, isto é, controlar e observar os nós dos elementos de memória (flip-flops,latchs...etc) de tal forma que no modo teste se pudesse enxergar apenas um circuito combinacional. Para tal, procurou-se seguir certas regras no projeto lógico pensando na utilização de técnicas, descritas mais adiante, de testabilidade.

Cita-se, a seguir, as regras utilizadas para a concepção do CI:

Foi evitada a utilização de circuitos contadores e registros assíncronos. A utilização de circuitos síncronos permite que os elementos de memória sejam mais facilmente testados através da implementação da técnica de "scan-test", descrita mais adiante, propi

ciando também um maior controle / observação da lógica combinacional. 255

O projeto dos blocos combinacionais não continha realimentações do tipo mostrado na figura 1. Este procedimento não permitiu o aparecimento de circuitos sequenciais gerados pela própria interconecção das portas lógicas.

Procurou-se evitar circuitos combinacionais de muitos níveis de portas lógicas. Os blocos de circuitos combinacionais eram separados por elementos de memória, conforme sugere a figura 2.

Para a introdução de dados nos flip-flops funcionais procurou-se a não utilização das entradas "sets" ou "resets" dos mesmos; portanto a entrada de dados em flip-flops deveria ser controlada pelo clock.

TÉCNICAS DE TESTABILIDADE IMPLEMENTADAS NO PROJETO:

Com o conceito de que elementos de memória em um C.I. podem ser chaveados para se transformar em um " Shift-register", pode-se controlar e observar os estados lógicos destes elementos. Foi, então, utilizada a técnica de "scan-test" parcial [7], acrescida de algumas técnicas de multiplexagem de sinais.

O "scan-test" consiste da utilização dos flip-flops funcionais do circuito, ligeiramente modificados, configurados na forma de um "Shift-register" quando no modo teste. Desta maneira podemos colocar e retirar dados internos ao circuito, de forma serial, sem que aumente em muito os pinos do CI. São necessários alguns pinos de controle para esta técnica. Obtem-se, então, um significativo aumento nos pontos de controle/observação internos do circuito. Existem dois outros aspectos importantes: o primeiro, é que o teste dos flip-flops se torna extremamente fácil, pois temos controle/observação diretos sobre eles; o outro, é que se pode decompor o circuito nas partes sequenciais e puramente combinacionais.

Foi visto que a geração de vetores de teste para circuitos puramente combinacionais é mais fácil que para circuitos mistos. Temos no exemplo da figura 3a um flip-flop comum, modificado para trabalhar no "scan-test"; na figura 3b observamos um pequeno circuito utilizando estes flip-flops modificados.

Quando nos referimos à técnica utilizada, "scan-test" parcial; isto quer dizer que nem todos os flip-flops funcionais do circuito foram incluídos no "scan". Esta ocorrência é devida ao fato de que nem sempre conseguiu-se seguir as regras estabelecidas para todas as partes do circuito, inviabilizando a implantação do "scan-test" nos elementos de memória destas partes. Apresentaremos mais adiante dados sobre o número de flip-flops envolvidos nesta técnica.

A técnica de multiplexagem de sinais também foi muito utilizada neste projeto, sanando diversas dificuldades na implementação da técnica de "scan-test".

Para permitir a entrada e saída de dados do "shift-register" quando no modo teste era necessário que se tivesse um clock comum para todos os flip-flops do "scan"; na impossibilidade de se ter um projeto lógico com todas as estruturas síncronas a um mesmo clock, optou-se pelo chaveamento, no modo teste, dos clocks das diversas estruturas síncronas a um único sinal, o clock-teste (CKT). Foram usados, então, Multiplex de 2:1 para este chaveamento.

Outra utilização importante dada à técnica de multiplexagem, ainda não encontrada na literatura, foi o chaveamento de nós de difícil observação da parte combinacional do circuito, para nós de observação direta via "scan-test". Isto implica na possibilidade de se observar diretamente dois nós do circuito com apenas um flip-flop pertencente ao "scan". Este procedimento nos levou à criação de mais um pino no CI, que é o controle deste chaveamento. Por outro lado, nos possibilitou a observação direta via "scan" de nós críticos no projeto para testabilidade.

Temos na figura 4 a estrutura básica utilizada nesta implementação.
Foram criados alguns pinos extras, não funcionais, para a utilização quando no modo teste; são pinos de controle do circuito de teste.

Existe no circuito um sinal particularmente importante, que é um pino de inicialização; ou seja, permite que se possa "setar " ou "resetar" todos os flip-flops do circuito. Pode-se conhecer, assim, no modo teste, o estado lógico inicial de todos os elementos de memória. Funcionalmente este pino é utilizado para o "power on reset".

RESULTADOS OBTIDOS:

Para análise dos resultados obtidos são feitas algumas considerações a seguir:

Os números de portas lógicas se referem a : 1 porta = 1 NAND de duas entradas, 1/2 porta = 1 inversor, 1 1/2 portas = 1 NAND de três entradas e assim por diante.

O tipo de falha considerado para a geração de vetores de teste e cobertura de falhas são do tipo simples amarrado em "0" ou "1"(stuck-at-"0" ou "1").

Os vetores de teste foram gerados manualmente com auxílio do programa SIMUFA [8] na verificação destes.

Os dados sobre cobertura de falhas ainda não são apresentados, atualmente, procede-se a uma simulação de falhas exaustiva, que estará pronta até a apresentação deste trabalho.

São mostrados na TABELA 1, dados do projeto e resultados obtidos:

* Nº total de portas lógicas do circuito	1500
* Porcentagem de lógica adicional p/ o projeto de testabilidade	19%
* Nº total de flip-flops (DFF)	85
* Flip-flops incluídos no "scan" (total)	62
* Flip-flops funcionais incluídos no "scan"	59
* Nº de entradas/saídas primárias (pinos)	26
* Nº de entradas/saídas via "scan"	139
* Pinos p/ controle do modo teste(não funcionais)	4
* Nº de vetores de teste via "scan"	73
OBS: Cada vetor é constituído de uma palavra de 62 bits	

CONCLUSÃO:

Projeto para testabilidade está se tornando uma necessidade nos projetos de CI's digitais complexos; muito se tem publicado e estudado a respeito, no entanto, encontra-se ainda grandes impendâncias por parte dos projetistas na utilização das técnicas para testabilidade.

Os resultados preliminares obtidos no projeto realizado no IM-CTI são considerados bons. Pelas características do circuito, altamente sequencial e com poucos pinos de controle /observação, não se visualizou, após o término do projeto,a viabilidade de teste do CI sem que se entrasse no modo teste. Isto mostrou claramente a importância da abordagem usada de projeto para testabilidade.

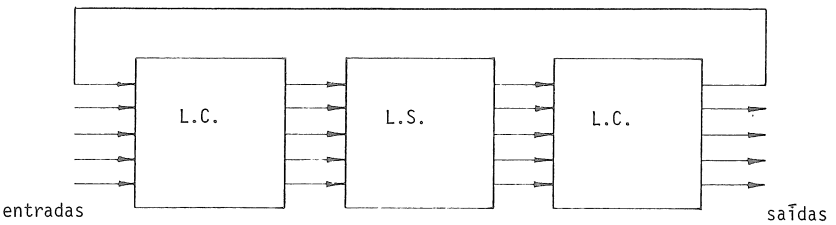


FIGURA 1

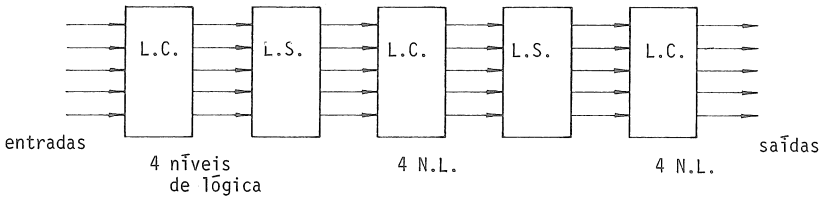


FIGURA 2

OBS: Para figuras 1 e 2 : L.C.- Lógica Combinacional
L.S.- Lógica Sequencial

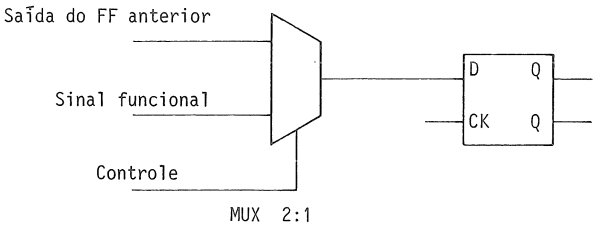


FIGURA 3a

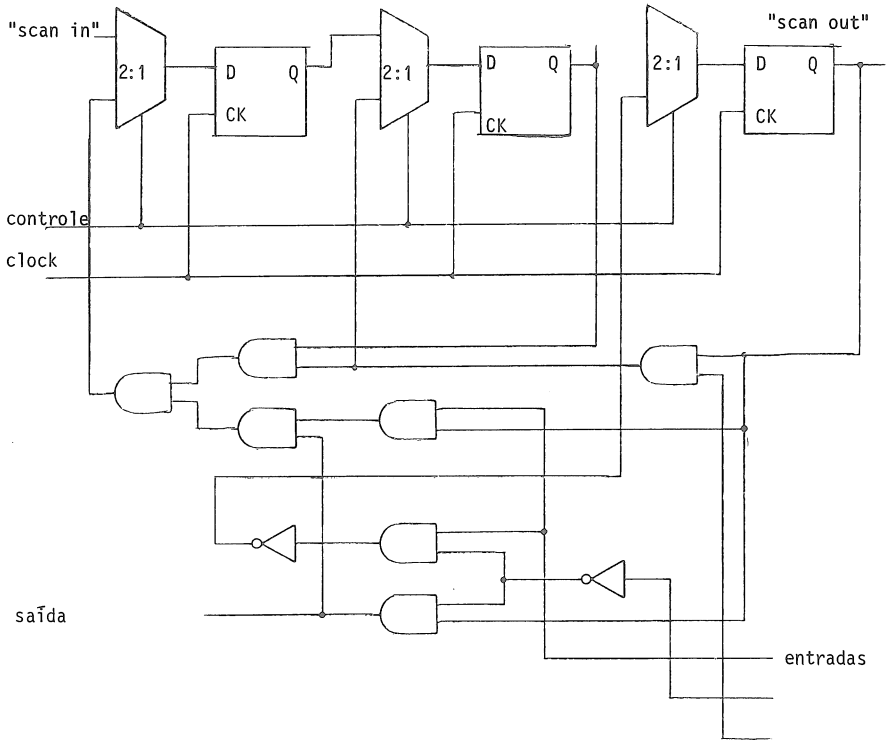


FIGURA 3b

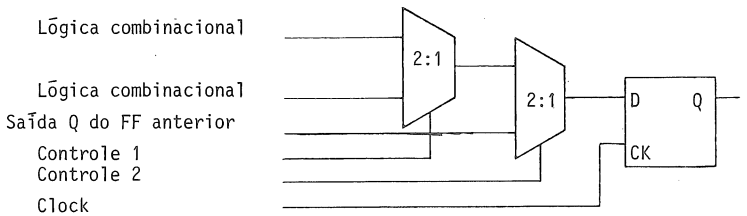


FIGURA 4

[1] Jacob A. Abraham, "Design for Testability", Custom Integrated Circuits Conference Rochester, N.Y., May 23-25, 1983.

[2] Thomas W, Williams and Kenneth P. Parker, IEEE Members, "Design for Testability- A Survey" Proceedings of the IEEE, vol. 71, nº 1, jan, 1983.

[3] Bennetts, R.G., "Introduction to Digital Board Testing".

[4] Akira Miyoshi, "Automatic Generation of Diagnostic Programs for Tosbac- 5400/150", Tokyo Shibaura Electric Company, LTD, Tokyo, Japan .

[5] Charles W. Cha, William E. Donath e Fusun Orguner, "9 V. Algorithm for test Pattern Generation of Combinational Digital Circuits"- IEEE Transactions on Computers, Vol. C-27, nº 3, March, 1978.

[6] Peter N. Marinos, member IEEE, "Derivation of Minimal Complete Sets of Test-imput Sequences Using Boolean Differences", IEEE Transcations on Computers, Vol. C-20 nº 1, Jan., 1971.

[7] Erwin Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology", Corporate Research & Technology, Cherry Hill, New Jersey 08034.

[8] R. Pannain, P.C.M. Freitas, R.A. Orives, M. de Moraes, "Ferramenta de Auxílio à Verificação de Vetores de Teste", Centro Tecnológico para Informática, Instituto de Microeletrônica, Campinas, S.P.

****

FERRAMENTA PARA TESTE DE CIRCUITOS VLSI UTILIZANDO UM MICROSCOPIO ELETRONICO

C. ORELLANA HURTADO*

SUMARIO

Este trabalho apresenta a princípio de funcionamento, descrição, principais técnicas de medida e aplicações de uma ferramenta de testes de circuitos VLSI usando um microscópio eletrônico de varredura aliado ao fenômeno do contraste por tensão. Esta ferramenta nos permite fazer uma análise pictográfica das imagens contrastadas correspondentes aos níveis de tensão na superfície do circuito. Também, podemos usar o feixe eletrônico como uma ponteira para medida de sinais internas de alta frequência. Finalmente são apresentados resultados experimentais de testes de circuitos efetuados no Curso de Pós-Graduação em Ciência da Computação da UFRGS.

ABSTRACT

This paper presents the functioning principle, description, measurement techniques and aplications of a VLSI circuits testing tool using a scanning electron microscope (SEM). This tool is based in the effect of voltage contrast. It allows pictorial analysis of the images contrast. Electron beam, can be used the measurement of high frequency signals in internals voltages nodes. Finally, some I.C. tests made at the Curso de Pós-Graduação em Ciência da Computação of the UFRGS are described.

* Engenheiro Eletrônico (UFRGS, 1982); aluno do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Rio Grande do Sul; Caixa Postal 1501 - 90.000 - Porto Alegre - RS - Brasil.

O crescente aumento do grau de integração de circuitos VLSI, assim como o aprimoramento de novas tecnologias visando aumentar a velocidade de propagação dos sinais e diminuição da potência consumida, faz-se necessário o desenvolvimento de equipamentos de teste mais versáteis e eficientes do que as ponteiras mecânicas. Na fase de desenvolvimento de um circuito integrado, medidas de sinais internos são sempre necessárias. É prática comum a introdução de ponteiras mecânicas, que em contato com as linhas metálicas, permitem obter em um osciloscópio as formas de onda dos sinais correspondentes.

O uso de ponteiras mecânicas nos traz um problema aparente: o risco de danificar o circuito. Mas, a principal limitação no uso destas ponteiras é a capacitância parasita inserida pela mesma. Esta capacitância pode introduzir erros na medida ou, ainda pior, pode causar o mau funcionamento do circuito. A tendência, como já mencionamos, do aumento dos fatores de integração (lei de Moore) nos permite conceber circuitos mais complexos tanto na sua estrutura como no seu funcionamento, tornando-se absolutamente necessária a medida das formas de onda dos sinais internos ao Circuito. Também, como conseqüência desta maior integração, teremos a diminuição das dimensões geométricas dos elementos e portanto, a diminuição das capacitâncias dos nodos internos. Isto torna mais evidente a limitação do uso das ponteiras mecânica-, no que tange ao seu posicionamento e a influência da capacitância parasita nas medidas.

O uso do SEM (scanning electron microscope) como ferramenta de testes, além de se apresentar como uma alternativa ao uso das ponteiras, nos permite a visualização dos níveis de tensão na superfície do circuito. Isto é, podemos fazer uma análise pictográfica através das imagens contrastadas da superfície. As principais vantagens da ferramenta SEM em relação as ponteiras mecânicas são:

- O feixe de elétrons pode ser posicionado precisamente sobre qualquer ponto da superfície, permitindo retirar sinais de elementos tais como: trilhas, transistores, contato, etc...
- O teste com o feixe é não destrutivo. A baixa energia de aceleração dos elétrons e a pequena corrente do feixe 10nA /ORE 84/ não causam danos de radiação. Não existe a chance de danos mecânicos.
- Rápido posicionamento do feixe. O posicionamento das ponteiras requer muito cuidado e é lento.
- A baixa capacitância apresentada pelo feixe 10E-17F /LUK 82/ nos permite a medida de sinais variando em alta freqüência e não afeta o funcio

namento normal do circuito.
- Permite a medida de vários sinais em paralelo.

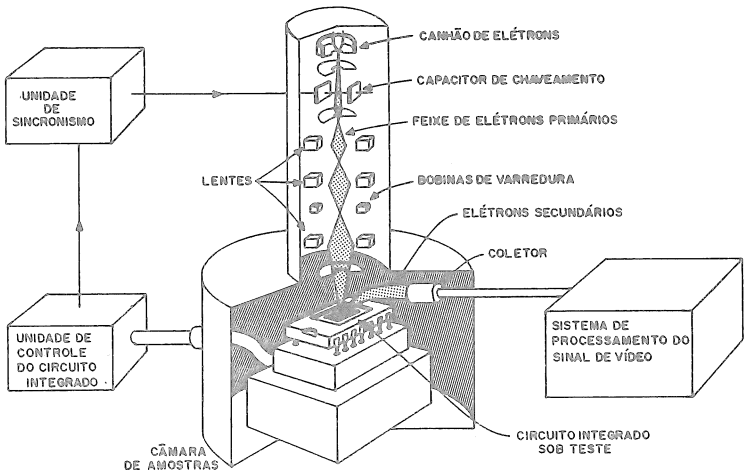


Fig. 1 - Ferramenta

2. FERRAMENTA

A configuração básica da ferramenta é um microscópio eletrônico de varredura (SEM). Podemos imaginar o sistema de observação do microscópio como sendo um circuito fechado de TV para observação da topografia de amostras. Desta forma, podemos visualizar a imagem da superfície da amostra no terminal de vídeo do equipamento. As partes que compõem o sistema de observação são: o sistema de varredura de feixe eletrônico e coletor, a câmara de amostras (estado de vácuo), a unidade de sincronismo e o sistema de vídeo. Através de chaves manuais no painel podemos controlar cada uma destas partes. O feixe é gerado no filamento e acelerado pelo canhão de elétrons. Depois de passar por estágios de condensação e alinhamento, o feixe é direcionado pelas bobinas de varredura de forma a varrer uma área retangular da amostra linha por linha; a freqüência de varredura é determinada por uma base de tempo previamente escolhida. A superfí-

cie atingida pelo feixe emite elétrons secundários, estes são detectados por um coletor de elétrons. O coletor esta opticamente acoplado a um fotomultiplicador, cuja saída está ligada a um estágio amplificador por onde o sinal passa antes de entrar no sistema de vídeo, onde o mesmo modula a grade de brilho do vídeo, obtendo assim, a imagem da amostra. O microscópio também pode funcionar no modo ponto, quando temos o feixe eletrônico livre do sistema de varredura, podendo posicioná-lo através de chaves manuais X e Y. As principais modificações para adaptar um microscópio a nossas aplicações são:

- Suporte de amostras adequado a colocação de um chip em funcionamento
- Atuação no feixe eletrônico para o chaveamento do mesmo
- Atuação no sistema de varredura para automatização do posicionamento do feixe.

Na figura 1 vemos a configuração completa da ferramenta. O teste de um CI inicia com a colocação do mesmo dentro da câmara de amostras, excitação das entradas a partir do bloco de controle, e posterior retirada de informação (imagem ou sinal) via feixe eletrônico.

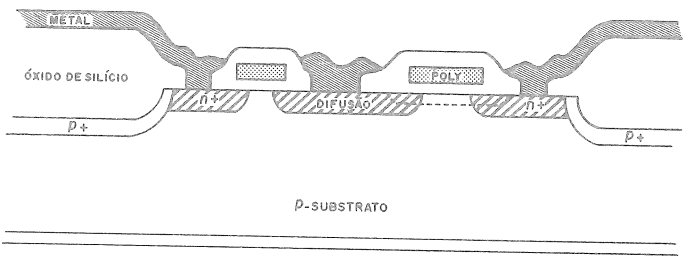


Fig. 2 - Vista de perfil de um inversor NMOS

3. PRINCÍPIO DE FUNCIONAMENTO

Dentro da câmara do microscópio eletrônico a superfície da amostra em observação é varrida pelo feixe de elétrons primários altamente energizados. A colisão destes com a superfície gera a emissão de elétrons secundários. Estes elétrons emitidos (secundários) são altamente sensíveis a campos eletrostáticos e são quase que totalmente coletados pelo campo formado ao redor do coletor. Cada ponto da imagem é proporcional ao número de elétrons secundários emitidos por cada ponto varrido pelo feixe. Os

contrastes das imagens mostram basicamente características da topografia da superfície. Na observação normal de uma amostra, a carga da superfície devido a chuva de elétrons, ocasiona a perda dos contrastes da imagem. Evita-se o problema com a deposição de uma camada fina de alumínio, de tal forma a podermos ligar a superfície a tensão mais positiva do sistema. Assim, para efeitos de análise da topografia, os contrastes observados nas imagens devem-se principalmente aos relevos da amostra. Para uma distribuição de diversas tensões na superfície, observamos imagens contrastadas segundo esta mesma distribuição: estamos então, diante da ocorrência do fenômeno do CONTRASTE POR TENSÃO.

O princípio deste fenômeno está no formação de pequenos campos elétricos ao redor das regiões polarizadas. Estes campos sobrepõem-se ao campo gerado pelo coletor (+250V) anulando seu efeito. Desta forma, os elétrons emitidos de regiões polarizadas com tensões diferentes da massa são submetidos a ação destes pequenos, mas atuantes campos elétricos, ocasionando um retardo dos elétrons até o coletor. Os elétrons emitidos de regiões da superfície ligados a massa são quase que totalmente coletados. Assim, como dissemos anteriormente, o sinal obtido na saída do amplificador de vídeo é proporcional ao número de elétrons coletados de cada ponto da superfície e portanto, ao nível de tensão destes pontos.

A topografia de um circuito integrado é composto por elemento tais como: interconexões, contatos, resistores, capacitores, transistores, etc ... Estes elementos são formados por diversas camadas de semicondutor, isolante e metal, como vemos na figura 2. A distribuição de tensões na superfície de um circuito integrado MOS digital é caracterizado pela existência de dois níveis referenciais de tensão 0V e 5V correspondendo aos níveis lógicos "0" e "1". A figura 3 mostra as linhas equipotenciais formados segundo a distribuição de tensões e a geometria do circuito. Os elétrons secundários emitidos das trilhas em ØV, deparam-se com um campo extrator através do qual chegam rapidamente ao coletor; já que os elétrons emitidos das trilhas em 5V passam por um campo de retardo antes de chegarem ao coletor ou são forçados a ficarem ao redor da trilha. Deste modo, obtemos um sinal proporcional que gera uma imagem contrastada no vídeo, com partes claras e escuras correspondendo aos níveis de tensão ØV e 5V respectivamente. As tensões entre ØV e 5V correspondem a partes cinzas de intensidade proporcional ao valor da tensão nesta região do circuito.

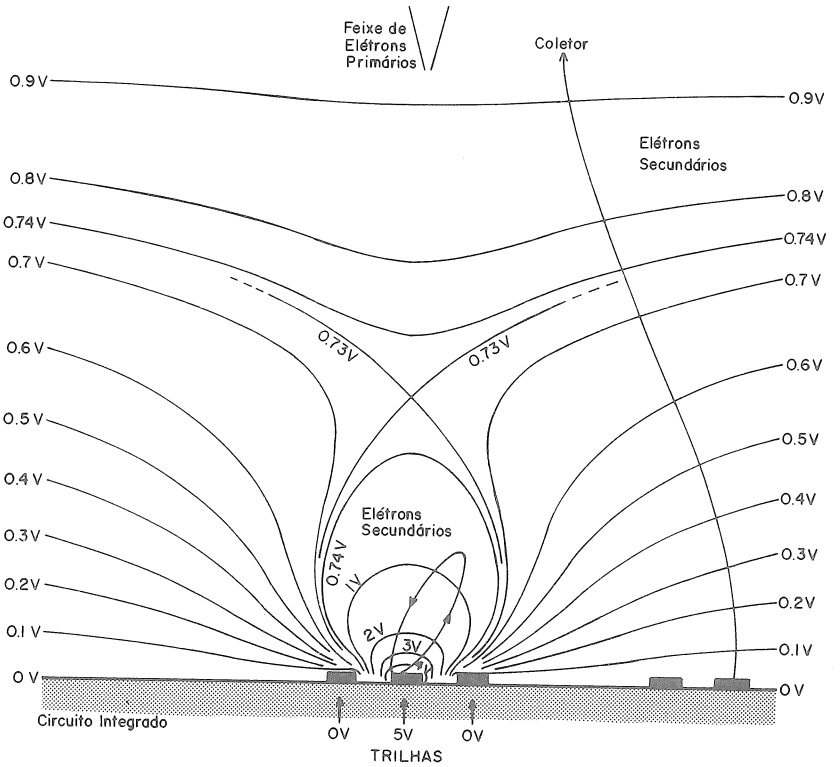


Fig. 3 - Linhas equipotenciais ao redor do circuito /FAZ 81/

4. FORMAS DE ANÁLISE E OBSERVAÇÃO

Há duas formas de análise e observação de CI's: estática e dinâmica. A análise estática realiza-se considerando uma situação estável dos estados de tensão do circuito em funcionamento. A obtenção de informações sobre os níveis de tensão na superfície é feita usando técnicas de medida que utilizam o feixe eletrônico contínuo. A varredura ao longo da superfície, ou posicionamento em um ponto do circuito, nos permite obter informações na forma de imagens contrastadas ou medidas quantitativas respectivamente. A análise e observação dos resultados nos permitem concluir sobre a correta polarização do circuito, linhas interrompidas ou em curto, identificação de elementos, identificação de caminhos de propagação de sinais.

A análise dinâmica realiza-se sob sinais dinâmicos do circuito em funcionamento. Desta forma podemos estudar circuitos funcionando na sua frequência de operação normal. As técnicas aplicadas para obtenção de informações utilizam o feixe eletrônico chaveado. A varredura ou posicionamento do feixe na superfície possibilitam obter informações na forma de imagens ou formas de onda. A análise e observação destes resultados nos permite concluir sobre o diagrama de tempos, obtenção de diagramas de estado, retardos e transição de sinais.

5. CHAVEAMENTO DO FEIXE

O chaveamento do feixe eletrônico é realizado com a mesma freqüência de operação do circuito em teste ou múltiplo desta, o mesmo nos permite a análise e medida de sinais variando em alta freqüência 1GHZ /HOS 78/. A implementação do chaveamento é feito através de um capacitor de placas como na figura 1 ou pelo canhão de elétrons /LUK 82/. Na figura 4 mostramos um sinal periódico sendo amostrado pelo feixe na transição, assim, através de sucessivas amostragens obtemos um sinal proporcional com a informação desta transição. A informação pode-se mostrar na forma de imagens ou formas de ondas, segundo a técnica de medida usada.

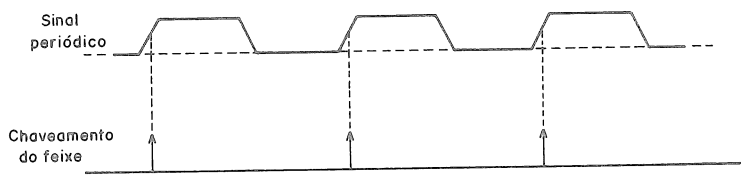


Fig. 4 - Chaveamento do feixe

6. TÉCNICAS DE MEDIDA

6.1 Contraste por Tensão Estática - RECORD

Esta é uma técnica simples e fácil de realizar. Aplica-se tensões DC nas entradas do circuito, estas resultam em uma distribuição interna de tensões e finalmente obtemos uma fotografia da superfície do circuito contendo os contrastes por tensão. A mesma é obtida através de uma câmera fotográfica acoplada ao sistema de vídeo do microscópio eletrônico. Na figura 5 podemos ver a fotografia da superfície de um circuito MOS com os

contrastes por tensão no metal, partes escuras indicando tensão 5V e partes claras ØV. O tempo de exposição ao feixe eletrônico é de 5s a 100s, tempo durante o qual o feixe varre a superfície do circuito linha por linha até completar um quadro. Através desta técnica a carga rápida da camada de óxido é evitada e consequentemente as fotografias obtidas mostram os contrastes por tensão característicos.

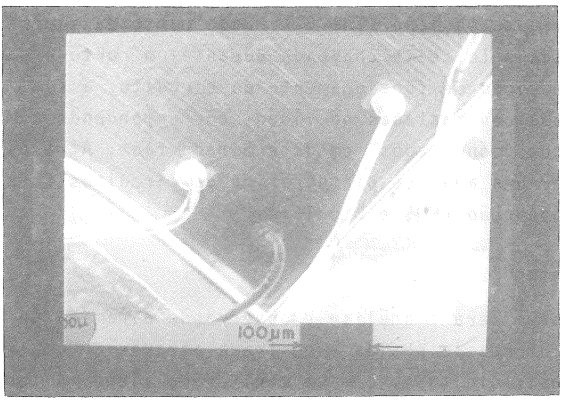


Fig. 5 - Contraste por tensão do C8748 (LAMM-UFRGS)

6.2. Contrastes por Tensão Estática - TV

Esta técnica nos permite obter no vídeo uma imagem com os contrastes característicos correspondendo a distribuição de tensões na superfície do circuito. A polarização da superfície do circuito é feita pela aplicação de tensões DC nas suas entradas. Também podemos aplicar sinais periódicos de baixa freqüência (<1HZ), para possibilitar o acompanhamento visual da mudança de contrastes, assim, poderemos ver a imagem da superfície com os contrastes fixos caracterizando tensões estáticas e mudança destes contrastes nas regiões onde existirem sinais variáveis. A imagem contrastada obtida no vídeo é como na figura 5. O problema enfrentado quando da aplicação desta técnica é a perda rápida dos contrastes da imagem, isto devido a carga da camada de óxido do circuito (overglass). A varredura constante do feixe carrega a camada de óxido. Para evitar este problema procura-se um balanceamento da corrente do feixe. O balanceamento é obtido pela escolha de uma aceleração de tensão ótima /WOL 79/, ou também, pela retirada da camada de óxido. A técnica pode-se aplicar em circuitos MOS e bipolar. Um circuito integrado MOS RAM não poderia ser analisado com esta técnica pois a frequência mínima de operação do cir-

cuito é 5ØØHz.

6.3 Técnica Estroboscópica

Esta técnica permite a análise e medida de sinais internos de alta frequência. Nesta técnica o chaveamento do feixe é realizado em sincronismo com a frequência de funcionamento do circuito. Para entender o princípio estroboscópico, podemos imaginar, por exemplo, uma haste girando com um período T, ao mesmo tempo uma luz piscando com o mesmo período ou T/n (n sendo inteiro), então observaremos a figura da haste não em movimento e sim fixa em alguma posição /SMI 81/. Cada ponto da superfície do circuito é amostrada pelo feixe chaveado durante um curto espaço de tempo em relação ao período de funcionamento do circuito, assim, teremos a formação das imagens contrastadas no vídeo, correspondendo a distribuição de sinais periódicos "congelados" em determinada fase. Através desta técnica podemos fazer uma análise pictográfica de circuitos funcionando na sua freqüência de operação (MOS e bipolar).

6.4 Voltage CODING

Esta técnica serve para a análise e observação dinâmica de CI's em funcionamento. A informação obtida é mostrada através de uma representação gráfica dos sinais digitais, com um monitor de vídeo. A técnica baseia-se na varredura do feixe chaveado ao longo de uma linha na direção Y do circuito, ao mesmo tempo, o sinal de vídeo correspondente a esta linha é distribuída na tela do vídeo em ambas as direções, obtemos assim, um gráfico dos sinais nas trilhas varridas pelo feixe, como na figura 6. A sequência de barras escuras e claras representa uma sequência de níveis lógicos variando ao longo do tempo na respectiva trilha, desta forma, podemos observar o estado lógico de um conjunto de trilhas. Exemplo, podemos observar os gráficos dos sinais em um barramento de dados ou endereços de um processador. O chaveamento do feixe é realizado em sincronismo com a frequência de funcionamento do circuito. A varredura do feixe ao longo do eixo Y do circuito pode ser realizado em qualquer frequência.

6.5 Técnica de Amostragem

Esta técnica permite a análise e observação de sinais digitais de alta frequência >10MHZ. Estes sinais não poderiam ser medidos por equipamentos tradicionais de medida devido a sua limitada largura de banda e a problemas causados pelas capacitâncias parasitas dos cabos. A técnica consiste na amostragem do sinal usando o feixe eletrônico. Conseguimos isto

através do chaveamento do feixe em uma frequência que é múltipla de frequência de operação do circuito. Desta forma, o sinal de um ponto qualquer do circuito é amostrado em fases diferentes, obtendo-se em cada amostragem o nível de tensão associado, que servirá para reconstruir a forma de onda deste sinal, como mostramos na figura 7. A frequência do sinal reconstruído é n vezes menor (n valor inteiro entre 1ØØ e 1ØØØ) /LUK 82/, possibilitando a análise do sinal por equipamentos cuja largura de banda é limitada. Para aplicação desta técnica é necessário a retirada da camada de óxido do circuito, de forma a usar uma corrente e aceleração de elétrons mínima. Existe a possibilidade de realizar a medida de sinais em pontos cobertos pela camada de óxido. Para isto aplica-se tensões de aceleração de elétrons da ordem de 1ØKV de forma a criar um canal de condutividade na camada de óxido (isolante) e assim retirar a informação do ponto /PEN 49/.

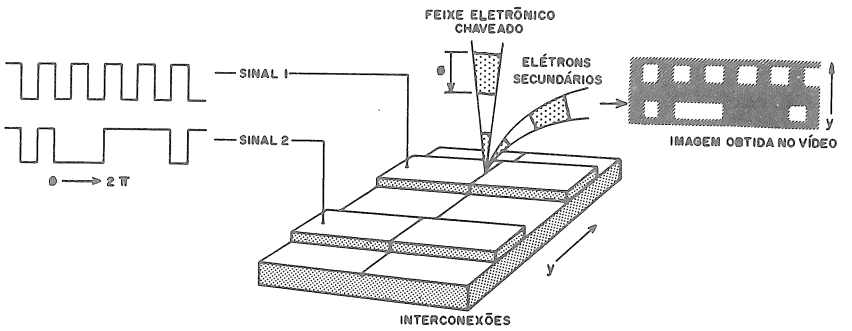


Fig. 6 - Voltage coding /FAZ 81/

6.6 Técnica MULTICANAL

Esta técnica nos permite a análise e medida de sinais periódicos em vários pontos do circuito ao mesmo tempo. O feixe de elétrons chaveado é posicionado em vários pontos sequencialmente, retirando de cada um a informação do sinal neste ponto, o sinal composto obtido na saída do amplificador do SEM é injetado em um analisador lógico, onde poderemos observar o timing de vários sinais como na figura 8. O número máximo de sinais a ser medidos ao mesmo é determinado pelo número de canais do analisador lógico. Esta mesma análise, se fosse feita através de ponteiras mecânicas, além de nos trazer riscos de dano mecânico e problemas de capacitâncias parasitas, precisaríamos de várias ponteiras.

7. APLICAÇÕES

7.1 Caracterização de Materiais

O SEM nos permite o estudo das características físico-químicas de semicondutores utilizados na fabricação de CI's. O feixe eletrônico induz correntes nos materiais atingidos. A análise destas correntes permitem concluir sobre os parâmetros físicos dos materiais.

7.2 Depuração de Protótipos

A visualização dos contrastes por tensão nas imagens nos possibilitam formas de depuração nunca antes imaginados, através das imagens podem detectar defeitos do processo de fabricação, bem como erros de projeto e serve também para controle de qualidade de circuitos. A inspecção pode ser realizado visual ou automaticamente, o processo automático se realiza através de processamento de imagens. Nas fig. 10 a 15 - vemos uma sequência de fotografias mostrando a descoberta e localização de um defeito devido a erro de projeto, o circuito testado foi uma pastilha contendo estruturas para análise de tecnologia, a estrutura oscilador em anel (cadeia de inversores) não funciona. A figura 9 mostra a topografia da estrutura sem polarização, na figura 10 observamos a estrutura com as linhas de alimentação polarizadas e a cadeia de inversores como uma sequência de pontos claros e escuros indicando estado lógico "Ø" e "1" respectivamente, esta sequência é interrompida na oitava linha (descoberta), através de sucessivas ampliações observamos que o problema é um transistor defeituoso (localização). Retirando a polarização da estrutura podemos ver na figura 14 o transistor defeituoso em detalhe e na figura 15 um transistor em bom estado. As fotografias foram obtidas no Laboratório de Microscopia Eletrônica e Microanalise (LAMM-UFRGS).

7.3. Análise de Falhas

A observabilidade do teste via SEM é muito bom, assim, a detecção de falhas, geração de padrões de falhas, obtenção de dicionário de falhas é realizado dispondo dos inúmeros pontos internos do circuito aos quais temos acesso.

7.4 Engenharia Reversa

A obtenção de esquemas elétricos de circuitos desconhecidos é muito importante para compreensão do seu funcionamento. Pode ser realizado manualmente pela observação das imagens do circuito ou automaticamente através de um sistema extrator de circuitos.

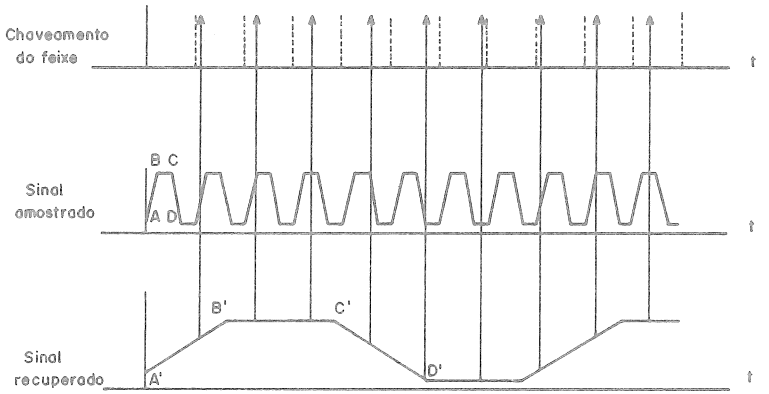


Fig. 7 - Amostragem do sinal

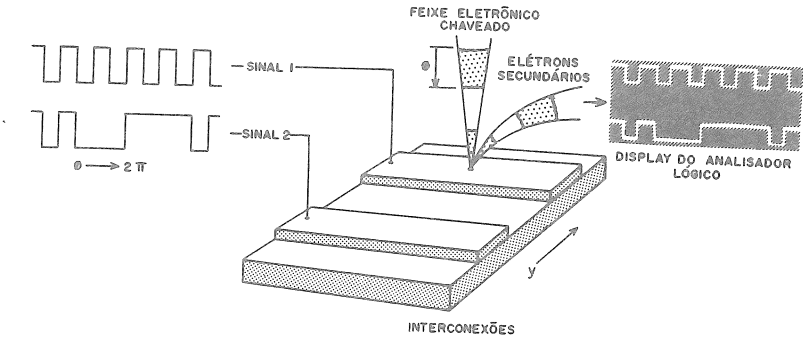


Fig. 8 - Multicanal /FAZ 81/

8. CONCLUSÃO

A definição desta ferramenta foi realizada como parte dos estudos preliminares para implementação da mesma. A implementação de uma ferramenta de testes usando o microscópio está sendo levada a cabo pelo Grupo de Microeletrônica do PGCC - UFRGS; como parte das primeiras experiências foram feitos testes em diversos circuitos e alguns são apresentados neste artigo. A chamada dos primeiros chips concebidos pelo grupo marcará o início da etapa de testes e depuração dos protótipos: a ferramenta SEM nos capacitará para tais empreendimentos.

9. AGRADECIMENTOS

Ao Prof. Dr. Altamiro Amadeu Suzim (PGCC-UFRGS) pela idéia inicial do trabalho e orientação no decorrer deste trabalho.

Ao Prof. Dr. Ricardo Reis (PGCC-UFRGS) pelas críticas, sugestões e fornecimento de material bibliográfico.

Ao Prof. Tiaraju Wagner (PGCC-UFRGS) pelas discussões e ajuda na implementação.

Ao Prof. Dr. Francisco Kiss (Departamento de Metalurgia-UFRGS) pelas facilidades concedidas na utilização dos recursos e sugestões.

Ao Sr. Sérgio Zin (Departamento de Metalurgia-UFRGS) pelos ensinamentos da área de microanálise.

Ao Prof. P. Jespers (Louvain-La Neuve, Bélgica) pelo fornecimento de pastilha de teste.

Ao pessoal da equipe de Arquitetura de Computadores do Laboratório TIM 3 (techniques de l'Informatique, des Mathématiques, dela Microélectronique et de la Microscopie quantitative) do Instituto IMAG-França pela bibliografia fornecida.

10. BIBLIOGRAFIA

/BAI 82a/ BAILLE, G. & LAURENT, J. Présentation d'un outil d'aide à la mise au point de circuits intégrées VLSI prototypes. Imag réport, Grenoble, Mars, 1982.

/BAI 82b/ BAILLE, G. et alii. Analysé de défaillances de VLSI par microscopie electronique. Imag réport, Grenoble, Juin 1982.

/BER 83/ BERGER-SABBATEL, G. & COURTOIS, B. E-beam testing strategies for VLSI. Imag report, Grenoble, Mai 1983.

/COU 82/ COURTOIS, B. Debugging VLSI using SEM. Imag report Grenoble, Sep. 1982.

/FAZ 81/ FAZEKAS, Peter et alii. Scanning electron beam probes VLSI chips. Electronics, New York, 54(14):105-12, July 14, 1981.

/FEU 78/ FEURBAUM, H. Quantitative Measurement with High Time Resolution of Internal Wave forms on MOS RAM's Using a Modified Scanning Electron Microscope. IEEE Journal of Solid - State Circuit, New York, 13(3):319-25, June, 1978.

/GOL 77/ GOLDSTEIN, J.I. et alii. Practical Scanning Electron Microscopy. Plenium Press, New York, 1977.

/GON 78/ GONZALES, A.J. & POWELL, M.W. Resolution of MOS one-transistor, dynamic RAM bit failure using SEM atroboscopic techniques. Journal Vacuum Science Technology, 15(3):1043-6, May/Jun. 1978.

/HOS 78/ HOSOKAWA, T. et alii. Gigahertz stroboscopy with the scanning electron microscope. Rev. Sci. Instrum., 49(9):1293-9, Sept. 1978.

/LAU 83/ LAURENT, J. & COURTOIS, B. Définition et Utilisation d'un outil de tests de VLSI par faisceau d'électrons. Imag report, Grenoble, Mai 1983.

/LUK 82/ LUKIANOFF, G.V. et alii. Electronics Beam Testing of VLSI Dynamic RAM's. Electronics Test, London, 5(6):46-56, Jun. 1982.

/OAT 69/ OATLEY, C. W. Isolation of potencial contrast in the SEM. Journal Phys. E.: Sci. Instrum., London, 2:742-4, 1969.

/ORE 84/ ORELLANA, C. Estudo preliminar de uma ferramenta de testes de CI's baseado no SEM. Porto Alegre, PGCC da UFRGS, 1984. (Trabalho individual).

/PEN 49/ PENSAK, L. Conductivity induced by electron bombardment in thin insulation fils. Phys. Rev., 75:472-8, 1949.

/PLO 68/ PLOWS, G. S. & NIXON, W.C. Stroboscopic scanning electron microscopy. Journal of Scientific Instruments, London, 1(2):595-600, 1968.

/SMI 81/ SMITH, K. Modified SEM depicts operation of dense chips. Electronics, New York, 54(13):73-4, June 30, 1981.

/WOL 79/ WOLFGANG, E. et alii. Electron - Beam Testing of VLSI Circuits. IEEE Transactions on electron devices, New York, 26(4):549-59, April 1979.

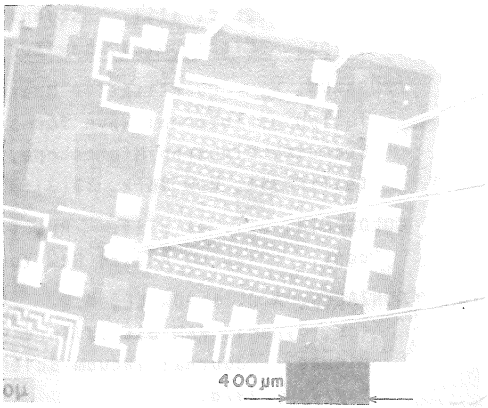


Fig. 9 - Fotografia do oscilador sem alimentação mostrando características topográficas (LAMM-UFRGS)

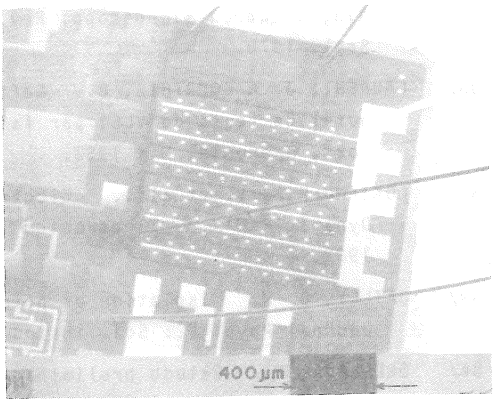


Fig. 10 - Fotografia do oscilador com alimentação mostrando o contrate por tensão nas linhas de alimentação (LAMM-UFRGS)

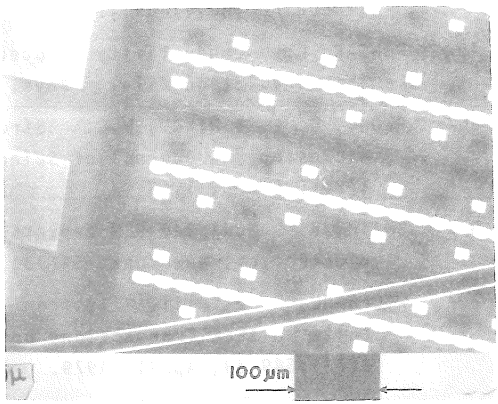


Fig. 11 - Fotografia do oscilador com alimentação mostrando a sequencia de partes claras e escuras interrompida.(LAMM-UFRGS)

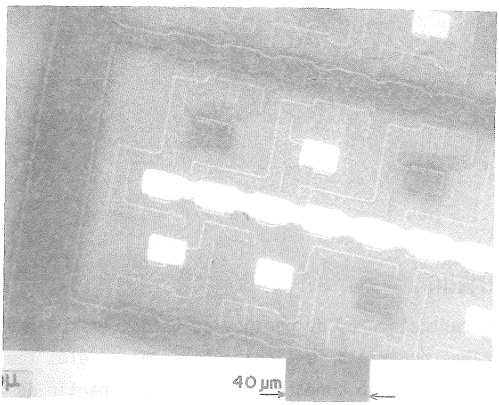


Fig. 12 - Fotografia do oscilador com alimentação mostrando o defeito (transistor defeituoso) (LAMM-UFRGS)

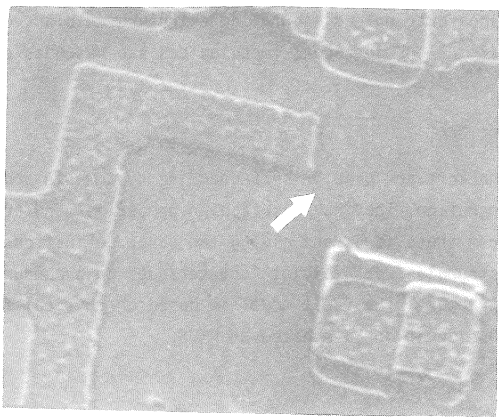


Fig. 13 - Fotografia do oscilador sem alimentação mostrando o transitor defeituoso (LAMM-UFRGS)

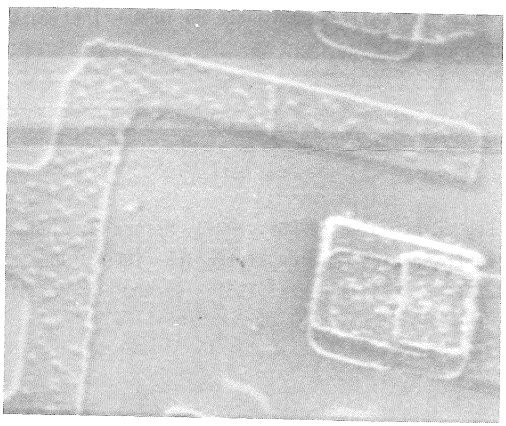


Fig. 14 - Fotografia do oscilar sem alimentação mostrando transistor em bom estado (LAMM-UFRGS)

CONCEPÇÃO CELULAR DE CONTROLADORES AUTOTESTÁVEIS BASEADA EM HIPÓTESES DE FALHAS FÍSICAS

I. JANSCH*

SUMÁRIO

Os controladores de código são usados em sistemas autotestáveis, visando a detecção de falhas durante o funcionamento normal - eles têm propriedades particulares e devem ser concebidos sob condições especiais. A partir deste pressuposto e das definições básicas desses circuitos, aborda-se os procedimentos de concepção dos controladores "de código fortemente disjunto" de estrutura celular, considerando-se modelos de falhas reais. As considerações tecnológicas são baseadas em NMOS.

ABSTRACT

Checkers are used in self-checking systems in order to detect failures during normal operation. They have particular properties and are designed under special conditions. From these ideas and giving the basic definitions of these circuits, we study the "strongly code disjoint" checkers design procedure, these chechers having a cellular structure. The fault model is based on physical fault hypotheses. The technological considerations are based on NMOS.

- Pesquisa realizada mormente no INPG - Instituto Nacional Politécnico de Grenoble, França.

* Eng. Eletrônica (UFRGS, 1979), Mestre em Ciência da Computação (UFRGS, 1982), Doutora-Engenheira em Microeletrônica (INPG, 1985). Endereço: Departamento de Informática/PGCC - Caixa Postal 1501 - 90.000 - Porto Alegre - RS.

I. INTRODUÇÃO

Desde que Galiay et al. /GAL 80/ mostraram os problemas decorrentes do modelamento de falhas através de "colagens" (do inglês "stuck-at"), o diagrama lógico revelou-se inadequado como representação de um circuito. Ao nível da concepção de um circuito e da geração de padrões de teste, é necessário usar modelos de falhas representando defeitos reais que podem ocorrer nos circuitos integrados. Estes modelos são referidos como "hipóteses de falhas físicas". O procedimento baseado no conhecimento da topologia real do circuito permite a detecção de circuitos abertos e curtos os quais, na prática, constituem-se na grande maioria das falhas físicas. Os sistemas autotestáveis aqui considerados são compostos de um circuito funcional e um controlador para os quais são requeridas propriedades bem definidas, como será visto resumidamente a seguir. Inicialmente definidos por Carter e Schneider /CAR 68/ e estudados por Anderson /AND 71/, os controladores eram definidos como "Totalmente Autotestáveis" (TAT) e "de Código Disjunto" (CD). Associados a um circuito funcional TAT (o qual é "Seguro a Falhas" (SF) e "Autotestável" (AT)), eles compõem um sistema completo que atinge o objetivo TAT, i.e., a primeira saída errada da rede é uma palavra não codificada. Os controladores de "código fortemente disjunto" (CFD) constituem a maior classe de controladores quando são considerados circuitos combinacionais. Eles são os companheiros naturais dos circuitos "Fortemente Seguros a Falhas" (FSF), definidos por Smith & Metze /SMI 78/.

Tais circuitos FSF formam a maior classe de circuitos funcionais, e admitem a ocorrência de falhas com relação as quais eles são redundantes. O conceito de redundância relacionado aos circuitos funcionais significa que seu comportamento para o espaço de código de entrada não é modificado em presença da falha. A "redundância forte" é uma extensão deste conceito para os casos em que ocorrem sequências de falhas. Então, os circuitos SFS mantêm-se seguros a falhas mesmo em presença dessas falhas.

Os controladores CFD são definidos /NIC 84/ com base em propriedades semelhantes: eles mapeiam entradas codificadas em saídas codificadas e entradas não codificadas em saídas não codificadas, mesmo existindo falhas internas.

Nicolaidis & Courtois /NIC 83/ analisaram a concepção de circuitos FSF baseada em hipóteses de falhas físicas. O objetivo deste artigo é o projeto de controladores com base nas mesmas hipóteses de falhas, as quais são englobadas sob a denominação da Classe I, segundo a definição de Courtois /COU 82/. Essa classe inclui defeitos como circuitos abertos, transistor MOS permanentemente aberto ou conduzindo, contatos ou pré-contatos falhados, curtos entre duas linhas de alumínio (respectivamente, duas de difusão) próximas geograficamente. Um único defeito físico é considerado, e as implementações físicas (ou regras dependentes da tecnologia) são baseadas na tecnologia NMOS.

Para todos os testadores aqui tratados, a indicação geral de erro é dada por um par de linhas de saída com valores complementares (código double-rail).

II. CONSIDERAÇÕES E DEFINIÇÕES BÁSICAS

Neste artigo só estão incluídas as definições diretamente relacionadas aos controladores CFD, tendo sido omitidas algumas intermediárias e outras referentes à teoria clássica e aos demais blocos componentes do sistema autotestável. Para entendimento desta estrutura global e análise dessas definições, sugerimos reportar-se às referências /COU 82/, /JAN 83/ e /JAN 85b/.

Consideramos os controladores como redes lógicas combinacionais (G) de múltiplas entradas e múltiplas saídas. O espaço de entrada X é composto de 2^r vetores binários de comprimento r, para r linhas de entrada. O espaço de saída Y é composto de 2^q vetores binários de comprimento q para q linhas de saída. Não ocorrendo falha, a rede G recebe somente um subconjunto de X denominado espaço codificado de entrada B e produz um subconjunto de Y denominado espaço codificado de saída C. Elementos que pertencem ao espaço codificado são ditos "palavras codificadas" (ou de código). Palavras não codificadas podem ser geradas sob a ocorrência de falhas.

A saída de G sob entrada x é representada por $G(x,f)$, para uma falha f presente na rede G, e $G(x,\emptyset)$, se não há falha.

O controlador de código fortemente disjunto é definido como se segue:

Definição: Um controlador é de código fortemente disjunto (CFD) para uma classe C_f de hipóteses de falhas se ele é de código disjunto e, se para todas as sequências de falhas f_i pertencentes a C_f que podem ocorrer, ou existe k tal que:

* o controlador é fortemente redundante com relação à sequência de falhas $\langle f_1, f_2, f_3, \ldots, f_{k-1}\rangle$ e
* $b \in B \mid G(b, \langle f_1, f_2, \ldots, f_k\rangle) \notin C$, ou
* o controlador é fortemente redundante com relação às sequências de falhas.

Lembramos que no caso dos controladores, a redundância com relação a uma falha dada pressupõe o mapeamento de entradas codificadas em saídas codificadas e de entradas não codificadas em saídas do mesmo tipo. O conceito de "fortemente redundante" é uma extensão do primeiro para o caso de ocorrência de uma sequência de falhas: o controlador deve ser redundante com relação a cada uma das subsequências $\langle f_1\rangle$, $\langle f_1,f_2\rangle$, $\langle f_1,f_2,f_3\rangle$, ...

Embora a Classe I considere apenas defeitos simples, a ocorrência sequencial de falhas individuais pertencentes a essa classe resulta em um grupo dessas, podendo ser visto como um defeito múltiplo. Essas falhas serão $\langle f_1\rangle$, $\langle f_1,f_2\rangle$, ... Cabe observar também que a sequência ... $f_i f_j$... não é igual a ... $f_j f_i$ ...

A seguir, passamos a analisar a concepção propriamente dita desses controladores CFD.

III. CONCEPÇÃO: OPÇÃO CELULAR

A concepção de um controlador de código fortemente disjunto não se constitui em uma atividade comum de projeto. O circuito e topologia resultantes devem assegurar a manutenção das propriedades dadas pela definição. Em /JAN 83/, mostrou-se um exemplo de topologia onde ocorre uma falha que não é detectada com a aplicação dos vetores de código, mas com relação a qual o circuito não é redundante. O resultado de uma situação destas é que, esta falha não sendo percebida, não é sinalizada. Mais tarde, ocorrendo uma outra falha no bloco funcional, (cujas saídas constituem-se em entradas do controlador) vai resultar em entradas não codificadas ao controlador. O controlador afetado pela falha não consegue mais garantir a propriedade de código disjunto, i.e., o fornecimento de saídas não codificadas, para este caso.

A complexidade da concepção desses controladores CFD está associada ao fato de que é necessário analisar o comportamento das propriedades desses diante de cada uma das falhas potenciais. Portanto, é evidente o aumento desta complexidade com o aumento do tamanho do circuito - a inclusão de cada nova linha exige estudo.

Como consequência desta constatação pensou-se na possibilidade de fazer o projeto baseado em células, nos casos em que é viável a estrutura celular regular do controlador. Uma vez concebida e verificada a célula de base, resta a examinar apenas as falhas que podem resultar da montagem destas, o que reduz enormemente o número de casos a serem estudados.

Não existem regras que garantam as propriedades CFD de células. Mas algumas delas podem servir como guias ao projetista do circuito, a fim de que ele possa conceber células novas para os casos ainda não disponíveis. O grau de dificuldade da montagem posterior pode ser diminuido pelo planejamento adequado das unidades.

IV. CASCATEAMENTO DE CÉLULAS DE CONTROLADORES CFD

A concepção de controladores de código fortemente disjunto pode ser feita a partir de células CFD, desde que essas sejam montadas de forma a preservar a propriedade CFD. Vimos que essas propriedades são definidas para:

* um conjunto ou classe de hipóteses de falhas;
* um conjunto de palavras codificadas (geralmente todas as palavras de código pertencem a um espaço de código de entrada;
* alguma hipótese de ocorrência de falhas com relação ao circuito funcional FSF e ao controlador CFD, assegurando que o objetivo TAT será atingido.

Essas propriedades devem ser definidas para um controlador concebido a partir de células CFD.

IV.A. Hipóteses de falhas

Ao realizar-se a montagem de células para compor um controlador, as falhas podem se relacionar às células e/ou às linhas usadas para interligar células. Isto é, essas falhas podem ser:

F1. falhas dentro de células;
F2. falhas afetando uma interconexão;
F3. curtos entre células;
F4. curtos entre interconexões;
F5. curtos entre células e interconexões.

As falhas possíveis que podem ocorrer dentro da célula são estudadas durante a sua concepção. A detecção dessas nas saídas da célula é assegurada a este nível e não é necessário revisá-las durante a montagem das células.

As falhas que afetam uma linha de interconexão dependem da classe de hipóteses de falhas. Sendo considerada a classe I, tem-se basicamente circuitos abertos. Estas falhas resultam em condições representáveis por "colagens", que podem ser facilmente detectadas com a aplicação de todas entradas pertencentes ao espaço de código de entrada, uma vez que estas linhas assumem os dois valores possíveis em condições normais. Portanto, devido a essa condição de colagem, uma palavra não codificada será rapidamente produzida nas saídas do controlador. Os circuitos abertos nas interconexões também podem ser vistos como transportados logicamente às entradas da próximo célula e esta condição também é estudada durante a concepção da célula.

Sendo considerada a Classe I, também os curtos entre células devem ser analisados. Esta verificação baseia-se na redundância do controlador com relação à falha resultante: se o controlador é redundante com relação ao curto, ou a falha pode ser detectada com a aplicação do espaço de código de entrada, esta não causará problemas. A maior dificuldade se refere aos curtos com relação aos quais o controlador não é redundante somente quando são aplicadas palavras não codificadas - assim eles não são detectados durante a operação normal do bloco funcional, e a sua possibilidade de ocorrência deve ser eliminada. Isto será visto mais tarde.

A detecção de curtos entre interconexões pode ser assegurada se cada par de linhas internas é ativado com os diferentes valores possíveis, assumindo a tecnologia NMOS. Particularizando esta situação para a classe I, e considerando somente os casos nos quais os curtos ocorrem entre linhas do mesmo nível (na estrutura em árvore), a detecção é assegurada se cada par de linhas internas vizinhas é ativada com os valores 01 ou 10. Outras situações como curtos envolvendo linhas de níveis diferentes na árvore, ou se o circuito não recebe um espaço de código de entrada completo, devem ser estudados como casos particulares.

No caso de curtos entre células e interconexões, nós consideramos linhas internas às células e não linhas conduzindo sinais de entrada e saída, uma vez que essas já haviam

sido consideradas no caso precedente. Se com a aplicação das palavras do espaço codificado de entrada ao circuito estas linhas recebem valores diferentes, esta falha será detectada, em geral. O mesmo problema de não-detectabilidade apesar de não ser redundante apontado no caso de curtos envolvendo linhas de células diferentes, também pode ocorrer no caso presente.

A situação ilustrada na figura 1 é a de um curto entre saídas intermediárias de um controlador de paridade par, ambos assumindo valores idênticos em operação normal, o que impede a detecção da falha. Tal curto impede o controlador de ser CFD. O problema pode ser evitado com a modificação do circuito de base ou com uma concepção cuidadosa (implementação física).

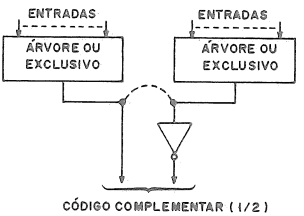


Figura 1: Curto-circuito não detectável em controlador de paridade

Outros exemplos serão dados com base em controladores "double-rail" (de código complementar). O circuito lógico /CAR 68/ e a versão elétrica correspondente, com número reduzido de transistores /NIC 84/, são mostrados pela figura 2. Se for feita a montagem de células "double-rail" cuja topologia corresponda ao da representação elétrica da figura 3, há o risco de ocorrer um curto entre o ponto P e a saída intermediária f_2, mesmo se separadamente estas células são CFD. Lembramos que o espaço de entrada do código double-rail se compõe de pares complementares, dois por célula (p.ex. X_1 e X_2, que podem assumir valores em 4 combinações possíveis). A falha assinalada não é detectável pelas entradas de código, e a concepção da célula deverá assegurar que o curto não será fisicamente possível.

IV.B. Palavras codificadas de entrada

Em geral, todas as palavras de código são necessárias para testar todas as falhas que podem ocorrer na célula CFD. Mas este pressuposto não implica que todas as palavras de código devam ser aplicadas ao controlador construído de células independentes, a fim de assegurar a detecção de todas as falhas passíveis de ocorrer.

Na seção precedente nós verificamos que há basicamente cinco grupos de falhas. Cada um destes tipos de falhas é testado por um conjunto de palavras de código, segundo descrição a seguir:

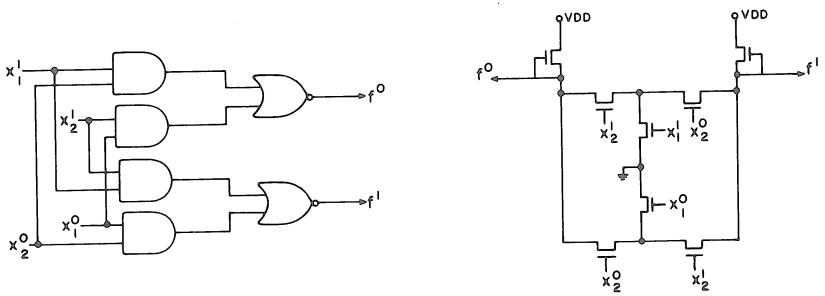


Figura 2: Controlador double-rail: circuito lógico e elétrico

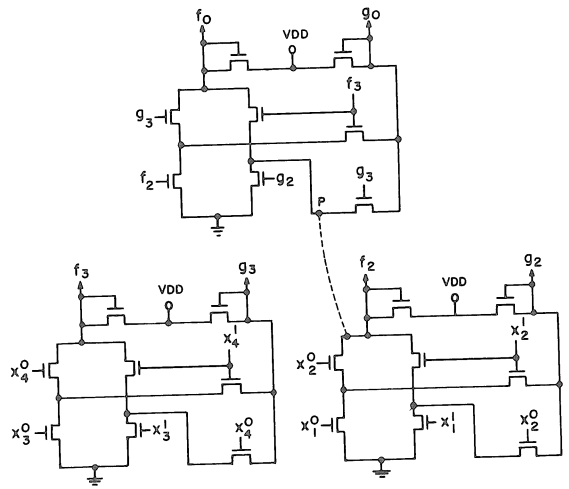


Figura 3: Curto não detectável em controlador celular double-rail

* as falhas detectáveis dentro das células são testadas com a aplicação do conjunto de palavras de código de entrada da célula;
* as falhas afetando uma interconexão são testadas pela aplicação dos dois valores lógicos diversos a cada uma destas linhas;
* os curtos entre células são testados com os valores de entrada do controlador que forçam uma das regiões afetadas a induzir um erro nas saídas da célula respectiva. Sendo esta condição atingida apenas com a aplicação de entradas não codificadas, o curto deve ser evitado por modificações topológicas;
* os curtos entre interconexões são testados com a aplicação das diferentes combinações de entrada a cada par dessas linhas (01 ou 10);
* os curtos entre célula e interconexão são testados à semelhança dos curtos entre células.

Assim, a fim de testar todos os tipos de falhas mencionadas, precisaremos, como palavras de código de entrada, o conjunto composto por aquelas descritas em todos os itens acima, que se constitui em um subconjunto do espaço de código de entrada (esse subconjunto e o espaço de código de entrada podem ser iguais ou não).

IV.C. Hipóteses de ocorrência de falhas

Faz-se necessário também considerar uma hipótese de falhas para o sistema - circuito funcional e controlador. A hipótese usual, com a qual pode-se atingir o objetivo TAT usando um sistema composto por circuito FSF e controlador CFD, prevê a ocorrência de falhas ou no circuito funcional ou no controlador, mas não em ambos em um mesmo intervalo de tempo. Isto nos garante que as situações de falha interna ao controlador são avaliadas independentemente das de erro em suas entradas. Ainda, concernente ao controlador, teremos apenas uma falha interna e outra não ocorre antes que todo o espaço de código de entrada seja aplicado. Esta hipótese está enunciada sob o nome de H2, em /JAN 83/.

Em certas proposições celulares, hipóteses menos restritivas podem ser consideradas. Estas possibilidades, entretanto, devem ser analisadas com referência a casos particulares de controladores e suas estruturas internas.

V. Procedimentos para a concepção e montagem de células

Regras para a concepção de células de controladores podem ser muito restritivas e não necessárias para todos os tipos de controladores. Mas elas podem auxiliar no projeto de novas células para diferentes códigos ou com outra forma e dimensões melhor adaptadas para aplicações específicas. A concepção ótima de uma célula pode facilitar o processo de montagem. Se as restrições não são dadas explicitamente com o projeto da célula, o usuário de uma biblioteca CFD pode necessitar de tempo e esforço não negligenciáveis para testar todas as falhas possíveis introduzidas pela reunião destas células e

assegurar a propriedade CFD do controlador resultante.

As regras apresentadas a seguir referem-se à topologia das células e interconexões e têm como maior objetivo evitar a ocorrência de falhas detectáveis unicamente pelas entradas não codificadas, causadas por uma dentre os três tipos possíveis de curtos descritos na seção IV.A.

A classe I de hipóteses de falhas não inclui curtos entre duas linhas de polisilício. Considera-se que tal defeito pode ocorrer apenas durante o processo de fabricação /COU 81/. Então, pode-se usar esta característica a fim de reduzir os defeitos possíveis - esta é a origem da regra R1.

Regra 1: Todas as interconexões entre células são efetuadas com linhas de polisilício. Essas interconexões aqui referidas são os sinais de entrada e saída de cada célula e não incluem linhas de alimentação.

A montagem de células CFD efetuada respeitando-se a regra R1, elimina as possibilidades de falhas envolvendo os curtos entre linhas de interconexão (ítens F4 e F5 da seção IV-A).

A possibilidade de curto-circuito entre duas linhas internas de células diferentes causando falhas não detectáveis pelo código de entrada (mas com relação às quais o circuito não é redundante) pode ser eliminada por modificações na topologia da célula, como sugerido a seguir.

Regra 2: As regiões limítrofes de cada célula que são concebidas em difusão devem corresponder a linhas de alimentação, V_{SS} OU V_{DD}. As linhas de metalização localizadas próximo aos limites da célula devem ser isoladas pela inserção de uma linha de alimentação em metal entre essas outras linhas e o limite da célula, se observa-se que um curto poderia ocorrer com uma metalização de outra célula que não seja linha de alimentação (ver figura 4).

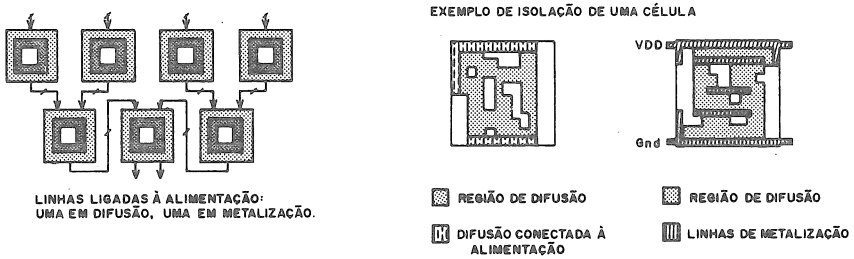


Figura 4: Restrições de isolação (regra 2)

A montagem de células CFD cujo projeto respeita a regra 2, interconectadas de acordo com a regra 1, e sendo assegurado que cada célula recebe seu espaço de código de entrada resulta um circuito controlador CFD, com relação à Classe I de hipóteses de falhas (sendo excetuados destas hipóteses os cortes nas linhas de alimentação).

O uso das regras 1 e 2 elimina tipos inaceitáveis de curtos descritos pelos ítens F_3, F_4 e F_5 (da seção IV.A), sendo considerada a classe I. Uma vez que os defeitos do tipo F_1 e F_2 são cobertos pelo uso de células CFD, o controlador completo é CFD.

A regra 2 é excessivamente restrita para alguns casos, mas ela pode ser particularizada para os diferentes códigos e estruturas de arranjo celular. Assim, para bordas de células situadas uma em frente à outra, apenas uma delas precisa respeitar a regra 2. Cumpre observar, entretanto, a associação implícita, neste caso, com restrições de lay-out - elas não podem ser, uma vez concebidas, associadas de qualquer maneira. As consequências sobre as propriedades CFD do circuito que resulta de operações de montagem não pré-definidas executadas com a célula - como rotações, translações, simetrias, etc... - devem ser analisadas previamente.

A restrição nas hipóteses de falhas mencionada na regra 2, pode ser eliminada se a regra 3 também for considerada.

Regra 3: Cada nível da estrutura em árvore deve ser alimentada por uma de suas extremidades, todas as linhas de alimentação do mesmo tipo tendo sua origem no mesmo lado. As linhas de alimentação dos diversos níveis devem ser separadas por outra de tipo diferente.

O uso da regra 3 resultará em uma organização esquemática semelhante a da figura 5. Isto significa que uma organização como a da figura 6 não é admitida. A sequência de defeitos ilustrada por $\langle f_1, f_2, f_3 \rangle$ resultará em um erro não detectável (mas não redundante) nas saídas. Logo, precisa ser eliminado.

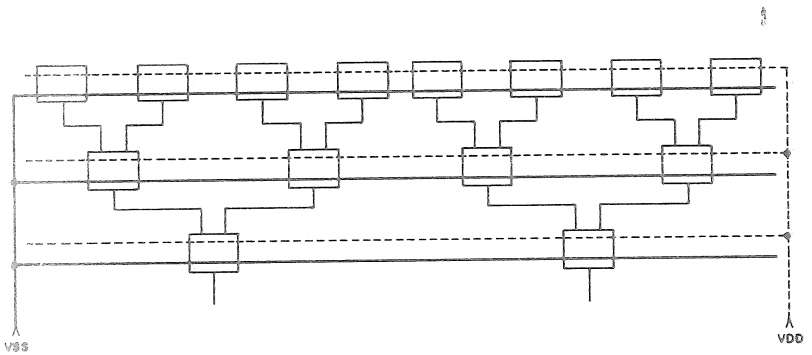


Figura 5: Linhas de alimentação organizadas de acordo com a regra 3

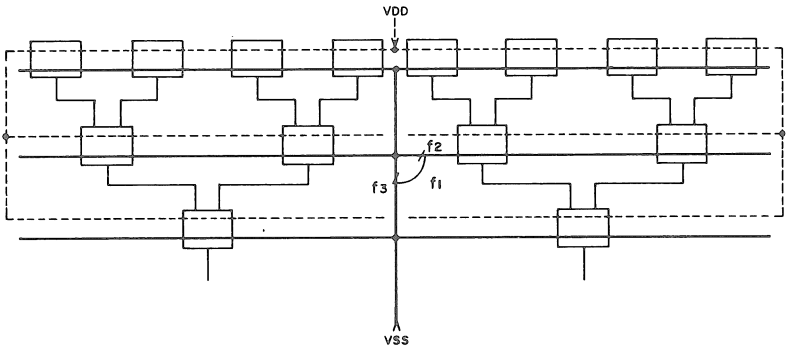


Figura 6: Sequência de falhas perigosas em linhas de alimentação

Uma montagem de células cuja organização e concepção respeita as regras R2 e R3, e cujas interconexões são feitas de acordo com a regra 1, assegurando que cada célula recebe seu espaço de código de entrada, resulta em um controlador CFD com relação à Classe I de hipóteses de falhas.

Conforme foi explicado no início, as regras aqui apresentadas são suficientes mas não necessárias. Outras menos estritas podem ser dadas, referindo-se a casos específicos de controladores. Elas encontram-se enunciadas e ilustradas em /JAN 85a/.

VI. CONCLUSÃO

Neste artigo, os controladores CFD são usados como componentes em sistemas autotestáveis - esses sistemas atingem o objetivo TAT, sob determinadas hipóteses de falhas.

Os principais problemas relacionados à concepção desses controladores, no que se refere ao caso de uma célula e à montagem da célula resultante, são discutidos. São dadas normas gerais para o projeto de uma célula, a fim de obter elementos convenientes à montagem, que possam compor uma biblioteca de células de controlador. Assim, o projeto de novos controladores pode ser facilmente obtido com o auxílio de ferramentas de PAC associadas.

REFERÊNCIAS

/AND 71/ ANDERSON, D. Design of self-checking networks using coding techniques. Univ. de Illinois, Urbana, Setembro 1971. 133p. (Relatório R527)

/CAR 68/ CARTER, W. & SCHNEIDER, P. Design of dinamically checked computers. Inf. Processing 68. Amsterdam, North Holland, 1969. v.2., p.878-83.

/COU 81/ COURTOIS, B. Failure mechanisms, fault hypotheses and analytical testing in LSI NMOS (HMOS) circuits" VLSI 81, Univ. de Edinburgh, Academic Press, 1981.

/COU 82/ COURTOIS, B. Analytical testing of regular circuits. In: IIº SIMPÓSIO BRASILEIRO DE MICROELETRÔNICA, São Paulo, 27-29 de julho de 1982. Anais. São Paulo, USP, 1982. (v.1). p.32-47.

/GAL 80/ GALIAY, J.; CROUZET, Y. & VERGNIAULT, M. Physical versus logical fault models MOS LSI circuits. IEEE Trans. on Computers, 29(6):527-31, Junho de 1980.

/JAN 83/ JANSCH, I. & COURTOIS, B. On the design of checkers based on analytical fault hypotheses. In: IIIº SIMPÓSIO BRASILEIRO DE MICROELETRÔNICA, São Paulo, 25-27 de julho de 1983. Anais. São Paulo, USP, 1983. (v.1, 1ª parte). p.41-56.

/JAN 85a/ JANSCH, I. Conception de contrôleurs autotestables pour des hypothèses de pannes analytiques. INPG, Grenoble, janeiro de 1985. (Tese de Doutor-Engenheiro).

/JAN 85b/ JANSCH, I. Circuitos autotestáveis: opção de projeto de circuitos visando o teste implícito ou concorrente. (proposto ao V Congresso da Sociedade Brasileira de Computação/SEMISH).

/NIC 84/ NICOLAIDIS, M.; JANSCH, I. & COURTOIS, B. Strongly code disjoint checkers. THE FOURTEENTH INTERNATIONAL CONFERENCE ON FAULT-TOLERANT COMPUTING. Kissimmee, 19-21 de junho de 1984. Anais. New York, IEEE, 1984. p.16-21.

/SMI 78/ SMITH, J. & METZE, G. Strongly fault secure logic networks. IEEE Trans. on Computers, 27(6):491-9, Junho de 1978.

Philip C. Treleaven and Safwat A. Mansi

Department of Computer Science
University of Reading
Reading RG6 2AX
England

ABSTRACT

The veritable explosion of current research into novel VLSI architectures is radically affecting the design of microprocessors and microcomputers. A whole spectrum of VLSI processors are under development based on the Mead-Conway design philosophy of simplicity, regularity and replication. These VLSI architectures range from special-purpose: including dedicated single processors such as Lyon's Optical Mouse chip and adaptable multi-processors such as Systolic Arrays, to general-purpose: including single microprocessors such as the Berkeley RISC and multi-processors such as the INMOS TRANSPUTERS. This brief survey of VLSI architectures will illustrate the exciting work going on in the area.

1. VERY LARGE SCALE INTEGRATION

Most of the technological achievements of the past decade have depended on microelectronics. Despite this the architecture of microprocessors and microcomputers has hardly changed. In the evolution of microelectronics we are now at very large scale integration (VLSI); in simple terms a chip containing over 100,000 devices.

However, VLSI technology has very different properties from the earlier microelectronic technologies. One important property of VLSI is the smaller feature size, giving benefits in higher density and speed, together with low switching energy and cost. These benefits are well understood [4,19]. In the evolution from 5 micron to 0.25 micron technology (by the end of this decade) we can expect an improvement in the switching elements in the order of two or more [15].

Another important property of VLSI technology is that the cost relationship between communication and switching has reversed [19]. Communication is more expensive and limiting in VLSI in at least four ways, as pointed out by Seitz [24]. Firstly, the wires occupy most of the area on a chip in contrast to transistor switches that rarely occupy more than a few percent of the area. Secondly, the delay in a Metal Oxide Semiconductor (MOS) logic element or driver is principally that induced on the element by the capacitance of the wire to be driven. Thirdly, the energy dissipated is likewise directly related to the capacitance and its dominant parasitic component. Lastly, the product of the resistance and capacitance per unit length of a wire scales so that only fairly short wires on a submicron feature size chip can be realistically regarded as delayless. We must therefore conclude that communication limitations are a fundamental and permanent characteristic of VLSI design and architectures. Thus high bandwidth communication is possible only between parts of a system that are physically close to each other.

In summary, the properties of VLSI are:

1. Design complexity is critical and requires careful design management based on simplicity, regularity and replication.

2. Wires occupy most space on a circuit and therefore equal attention must be given to the design of wiring as to components.

3. Non-local communication degrades performance, becoming progressively expensive in chip area, signal energy and propagation.

These properties of VLSI technology, notably the sheer complexity of utilising a hundred thousand logic gates, has led to the Mead-Conway VLSI design and implementation philosophy. Their VLSI philosophy has three important constituents. Firstly, the Mead and Conway VLSI design style documented in "Introduction to VLSI Systems" [19]. Secondly, the Multiproject chip concept [4] which places a number of separate designs on a single chip so as to reduce implementation costs. Lastly, the so-called Silicon Foundry [12] able to fabricate small quantities of designs from different sources, operating VLSI implementation like a photographic film processing service for its customers.

In the context of the above properties of VLSI and the Mead-Conway philosophy, the remainder of this paper examines the special-purpose and general-purpose architectures designed to exploit VLSI.

2. MACHINE ARCHITECTURES

Having briefly examined the properties of VLSI we are now in a position to list the criteria, and their related advantages, of "good" VLSI architectures.

Firstly, the architecture must be implementable by only a few different types of simple cells and simple chips. Since most of the cells and chips are copies of a few basic ones, only a few different simple cells need to be designed and tested. Kung [13] states that exactly how simple a cell should be can only be answered on a case by case basis. For instance, if a whole system is to be implemented on one chip, then each cell would probably contain only simple circuits and a small amount of memory, whereas for board implementations it is likely that each cell may approach the complexity of a simple microcomputer.

Secondly, the data and control paths of the architecture should be simple and regular. Cells may be connected by a network with local and regular interconnections, thus avoiding or minimising long distance or irregular communication. Regular interconnection implies that the design can be made modular and extensible, so a large computer system can be implemented by combining the designs of simple cells and simple chips.

Thirdly, the architecture's communication should be localised. Ideally there will be no global communication, only communication between adjacent cells and chips. This may be viewed as "locality of reference" applied to VLSI.

Fourthly, the architecture must utilise extensive concurrency for performance. High performance is obtained in VLSI architectures from the concurrent operation of a large number of simple (cells and chips) processors rather than from a single large processor. This concurrency may be obtained either by pipelining the stages involved in a computation or by multiprocessing independent computations in parallel.

Lastly, the architecture should make multiple use of each input data item. To balance input/output bandwidth with computation, it is necessary to make multiple use of input data. As Kung [13] has pointed out, this can be achieved either by broadcasting the input data to all cells, or by having the data traverse a regular structure of cells.

A whole spectrum of processor architectures is under development based on this VLSI philosophy of simplicity, regularity, and replication. These architectures range from special-purpose: including dedicated single processors such as Lyon's Optical Mouse chip [17] and adaptable multiple processors such as Kung's Programmable Systolic Array chip

[5], to general-purpose: including single microprocessors such as the Berkeley RISC [20,21] and multi-processors such as the INMOS TRANSPUTERS [1].

An interesting taxonomy [24] for this spectrum of VLSI architectures is proposed by Seitz. This is shown in Figure 1. The horizontal axis shows the "Processor" size which ranges from a replication element as small as a RAM or shift register cell, to a large programmable processor. The vertical axis shows the number of (often identical) processors that make up an architecture.

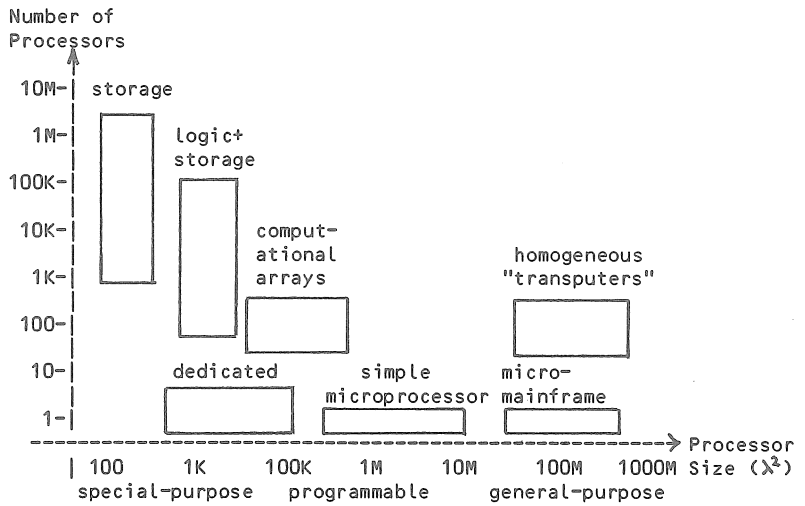


Figure 1: Taxonomy of VLSI Architectures

In a tour of these zones from left to right, one starts with Storage systems, such as random-access memories and shift registers, in which the replication cell is about as small as usefully imaginable. Next are what Seitz calls Logic+storage systems [24], systems in which a small amount of logic is associated with a few hundred bits of storage. A structure made attractive by the logic and storage being implemented in the same technology. Example applications include sorting and association etc.

In the next zone are Dedicated processors, single special-purpose chips built for a specific application such as the RSA cipher chip [23]. These processors have the advantage of a wired-algorithm so-to-speak. Next comes Computational Arrays which perform highly concurrent numerical computations. The processors are connected in regular

patterns that match the flows of data and control in the target computation, and usually are capable of operations such as addition and multiplication. Typical applications are signal and image processing.

To the right come the general-purpose processors. Zones with single processors: such as simple microprocessors based on the reduced instruction set concept, and Micromainframes [8]. The distinction between these two zones is partly one of programming style and partly one of engineering. Next comes zones with multiple processors, including networks of "transputer" microcomputers. The expectation for these multi-processor systems is that computations can be distributed across a system by a compiler and loader so that potential concurrencies can be exploited at execution time.

Below we examine examples of these special-purpose and general-purpose VLSI architectures to illustrate the exciting work.

3. SPECIAL-PURPOSE ARCHITECTURES

A dedicated special-purpose VLSI architecture is designed either to form an integral part of a digital system or as a peripheral device to be attached to a conventional host computer. It can be a single processor chip built from a replication of simple cells, or a multi-processor system built from identical simple chips, or even a combination of these two approaches. Special-purpose processors have a number of advantages [26] that help reduce their cost:

1. Only a few different simple cells need to be designed and tested since most of the cells and chips are copies of a few basic models.

2 Regular interconnection implies that the design is modular and extensible, so one can create a large processor by combining the designs of small cells and chips.

3. Many identical cells and chips that use pipelining and multiprocessing can meet the performance requirements of a special-purpose processor.

Besides the excellent special-purpose architectures below, many equally novel processors are found in the literature [7,14,22].

3.1. Single Processors

The two examples of special-purpose single processors briefly examined below are the RSA Cipher chip [23] and the Optical Mouse chip [17].

RSA Cipher Chip

This chip, designed by Ronald Rivest and his colleagues at the Massachusetts Institute of Technology, implements a public-key encryption algorithm. This operation is computationally demanding, requiring up to several hundred multiplications of several hundred bit numbers. The chip was designed as a general-purpose, big-number processor, using a bit-slice architecture for the ALU. It is a good example of a chip design based on the duplication of simple cells.

Externally, the chip is configured as a memory chip that can be read or written at one of four eight-bit word positions. For instance, one of these positions is the "window" for data I/O, while another is for receiving commands such as "encrypt." To support encryption, the RSA chip has a 512-bit ALU organized in a bit-slice manner. It has eight general-purpose 512-bit registers, as well as up-down shifter logic and a multiplier. Other sub-systems include control logic (containing a PLA of 224 72-bit microcode words), a stack/counter array for subroutines and loops, and an array of powerful superbuffers to drive the signals that control the ALU.

Internally, the ALU is only capable of performing the operations $(A * B) \pm C$, shift-left, shift-right, and test least-significant bit. The remaining required operations are implemented by microcode control subroutines. These operations include: RSA encryption/decryption (modular exponentiation), generating a large prime number, generating a complete RSA key-set, greatest common divisor, and input or output of a large number through the eight-bit window.

The "floor plan" of the RSA cipher chip is shown in Figure 2. The left side contains a block of 320 slices of the ALU and the upper-right block contains the remaining 192 slices. The central spine carries control signals to the ALU from the superbuffer driver array at its lower right. The microcode PLA occupies the right-center area of the chip, and the remaining logic (stack, pads, etc.) occupies the lower-right portion. In Figure 2, S denotes a stack, X an eight-bit "window," C a small bus-control PLA, and D some debugging logic.

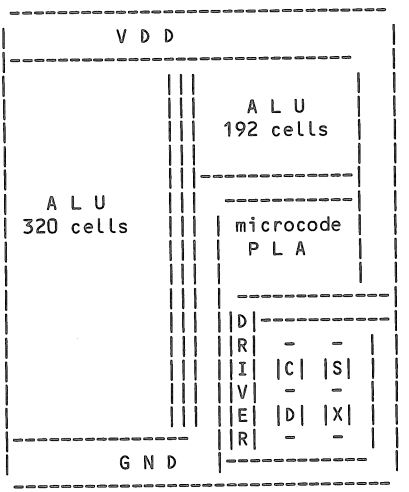


Figure 2: RSA Cipher Chip Floor Plan

The entire chip contains about 40,000 MOS transistors, has 18 pins, runs at 4MHz, and uses a little more than one watt of power. Rivest estimated that the project required about five man-months of effort, with the chip finally being fabricated in the fall of 1979. It was primarily a programming project because he and his colleagues almost exclusively wrote programs in LISP, which when executed, created the desired Caltech Intermediate Form output file. Altogether, they wrote about 75 pages of LISP code. The largest pieces specified the final placement and interconnection of the modules (14 pages), the description of the ALU (11 pages), and the microcode assembler and simulator (10 pages). Estimates are that the chip can perform RSA encryption faster than 1200 bps (even faster if shorter keys are used), and the designers predict speeds of 20,000 bps within a few years.

Optical Mouse Chip

Dick Lyon investigated [16, 17] new VLSI architecture methodologies for applications such as signal processing and smart digital sensors. A novel example of the latter is the Optical Mouse chip, designed in the VLSI system design area at Xerox Corporation's Palo Alto Research Centre.

The Optical Mouse is a pointing device for controlling the cursor on a personal workstation display, such as the

one found on the Xerox Star 8010 information system. The design was motivated by the desire for a highly reliable mouse with no moving parts except button switches, and it was realised through the innovative use of electro-optics, circuit design, geometric combinatorics, and algorithms - all in a single special-purpose sensor chip.

This chip reports the motion of visible spots relative to its coordinate system by combining two techniques. One is a simple "mostly digital" circuit that produces digital image (bitmap) snapshots of bright features on a dark field using self-timed circuit concepts and mutually inhibiting light sensors. The other technique uses a tracking algorithm with an easy-to-track contrasting pattern, a detector array and inhibition network matched to the pattern, and a digital machine that inputs images of that pattern and track relative image motion. (An especially interesting aspect of the design is the integration of sensors, memory, and logic in a single array using standard MOS technology).

The basis of the sensors is that in nMOS, light striking the circuit side of the chip converts photons to hole-electron pairs; the holes are attracted to negative-biased p-type silicon substrates, while the electrons are attracted to n-type regions. A so-called dynamic node that has been positively charged will detect light by collecting a negative charge (electrons) and "leaking" to a lower voltage. An imager is simply an array of subcircuits, each consisting of a dynamic node, a transistor to reset the node to "high" and isolate it, and an inverter circuit to sense the voltage of the node and communicate it to other circuits.

Figure 3 shows the floor plan of the optical mouse chip. "Mouse cells" represent the four-by-four, two-dimensional sensor array, and "Tracker PLA" tracks spots by comparing images and outputting X and Y movements, as well as outputting "Any-Good" and "Jump" counter control and test signals. The "X counter PLA" and the "Y counter PLA" control up/down counting and transmit "XA XB XL" and "YA YB YL" coordinates to the host computer.

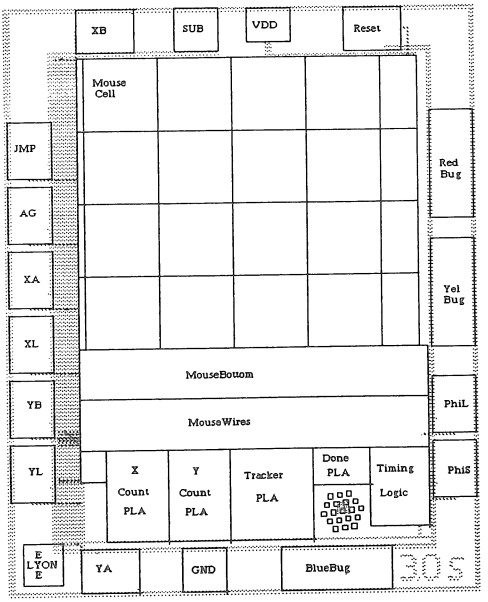


Figure 3: Optical Mouse Chip

The layout style used in this first version of the chip treats a sensor cell with its logic and memory as a low-level cell and constructs the array by selectively programming the cells in different positions. The resulting chip is about 3.5mm x 5.4mm in a typical nMOS process (Lambda=2.5 microns, or five-micron lines). A second version of the chip has also been designed to improve sensitivity.

3.2. Multi-Processors

Special-purpose multi-processor architectures are constructed from identical specialised processors. Some of the processors are dedicated logic, others are programmable to some degree. The processors are typically capable of operations such as addition and multiplication, and are connected in regular patterns that match the stream lines or wave fronts of the calculation. This is one of the most rapidly expanding areas of VLSI architectures because it remains one of the few proven multi-processor approaches for exploiting VLSI. The two examples examined here are the Geometry Engine [3] and the Programmable Systolic Chip [5].

Geometry Engine

This vector function unit, designed by James Clark when at Stanford University, performs three of the common geometric functions of computer graphics: transformation, clipping, and scaling. A single-chip version is used in 12 slightly different configurations to accomplish 4x4 matrix multiplications; line, character, and polygon clipping; and scaling of the clipped results to display device coordinates. This unit is an excellent example of a special-

purpose device built from identical chips.

When configured as a four-component unit, the Geometry Engine allows simple operations on floating-point numbers. Each of its four identical function units has an eight-bit characteristic and (currently) a 20-bit mantissa. It operates with a simple structure of five elements: an ALU, three registers, and a stack. This basic unit can perform parallel additions, subtractions, and other similar two-variable operations on either the characteristic or the mantissa. Since one register can shift down and one can shift up, it can also multiply and divide at the rate of one step per microcycle. The 12-chip system consists of 1344 copies of a single bit-slice layout composed of the five elements. Four pins on the chip are wired to indicate to the microcode which of the 12 functions to carry out, according to the chip's position in the subsystem organization.

Figure 4 shows the geometry subsystem described above. The terms "MM", "CLIP", and "SC" denote matrix multiply, clipping, and scaling. Each of the four "MM" blocks, for instance, is a separate Geometry Engine chip that does a four-component vector dot product. Although each multiplication is done at the rate of one partial product per microcycle, the matrix multiplier has 16 of these products simultaneously active. Thus, the total transformation time, which is the bandwidth limiting system operation, is about 12 microseconds.

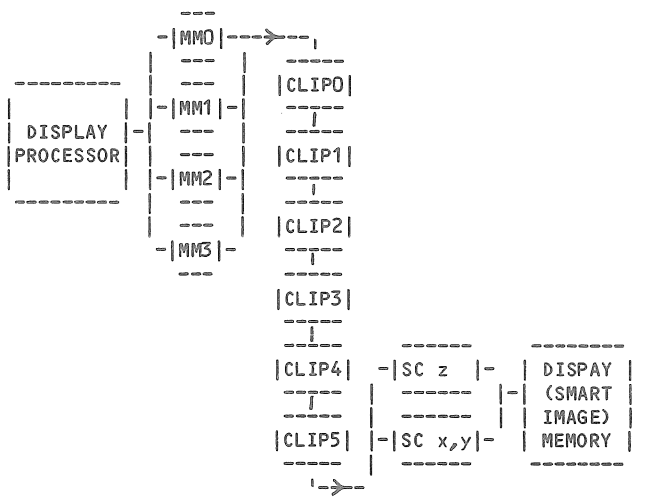


Figure 4: Geometry Engine data flow

In the actual design of the Geometry Engine almost all of the complexity was in the microcode that drives it. This microcode represents the logic equations for its finite-state machine, which are implemented in a PLA. Writing this microcode and making minor additions to the principal bit-slice to accommodate it took up approximately 50 percent of the total design time. Estimations are that the Geometry Engine is capable of performing about four million floating-point operations per second; 48 identical units, four per chip, will each do a floating-point operation in about 12 microseconds.

Programmable Systolic Chip

H. T. Kung and his co-workers at Carnegie-Mellon University are investigating the relationship of algorithm design to special-purpose chip architecture. A notable outcome of Kung's work has been the concept of "systolic arrays" [13]. Recently various computationally demanding problems have been solved using systolic algorithms. These systolic algorithms generally consist of simple processors, or systolic cells, that are connected in a regular pattern and operate on continuous, regular flows of data. To overcome the limited scope and hence the high design cost of a specialised systolic cell, the Programmable Systolic Chip (PSC) [5] has been designed to support many systolic algorithms. The PSC is tailored to the I/O and computational requirements of the family of systolic algorithms, so a large number of these chips can be used to implement a broad spectrum of algorithms.

A PSC processor [5] consists of five functional units (see Figure 5) that operate in parallel and communicate simultaneously in a pipeline fashion over three buses. The five functional units are: a 64x60-bit microcode dynamic RAM and a microsequencer, a 64x9-bit words DRAM register file, an ALU, a multiplier-accumulator (MAC), plus three input and three output ports. Data communication among the units takes place over three independent buses; control and status lines are separate. Each bus can be written by one of eight sources and can be read by any of ten destinations.

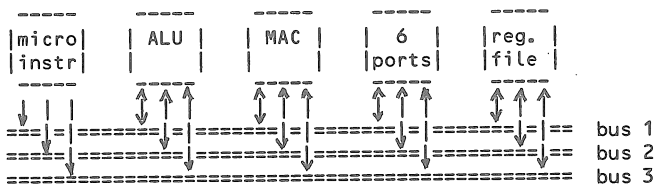


Figure 5: Programmable Systolic Chip

In order to keep the chip small and hence keep pinout and yields reasonable, an eight bit word size was chosen. Most data paths are, however, nine bits. The ninth bit being used to tag data, for control information, or as the most significant bit of a number modula 257 for coding applications. Each of the three input and three output ports are also nine bits; eight for data and the ninth for data or control. To keep the clock period short, interchip communication is pipelined with instruction execution. Thus a value which is computed in a given cycle can be used on the same chip in the next cycle, but not until the cycle after that on a neighbouring chip.

The PSC project, initiated in October 1981 has already led to a number of nMOS PSC processors being fabricated and tested. In the PSC the VLSI architecture problem of parallel programming is cleverly circumvented by limiting the programs supported to systolic algorithms and placing identical microcode in the DRAM of each systolic cell.

We will now consider general-purpose VLSI architectures.

4. GENERAL-PURPOSE ARCHITECTURES

Traditionally, the trend in designing microprocessors, and even mainframe computers, has been towards the use of increasingly complex instruction sets and associated architectures. However, the judicious choice of a simple set of instructions and a corresponding simple machine organisation can achieve such a high instruction rate that the overall processing power can exceed that of processors implementing more complex instructions.

In addition, a recent trend is for these simple processors to be designed so they can not only be used alone, but also can operate together as a parallel system.

Besides the interesting general-purpose architectures below, other novel processors are to be found in the literature [7, 9, 10, 25].

4.1. Single Processors

The two examples of general-purpose single processors briefly examined here are the MIT Scheme Chip [2, 11] and the Berkeley RISC [20, 21].

MIT Scheme-79 Chip

Two Scheme microprocessors namely Scheme-79 [11] and Scheme-81 [2] have been designed and built. Below the Scheme-79 chip which is more readily accessible in the literature is described. Notably, the Scheme-79 chip is a

"landmark" of the Mead-Conway VLSI design philosphy, taking just five weeks to design and layout.

The Scheme-79 single-chip microprocessor, developed by Gerry Sussman and his colleagues at MIT directly interprets a typed pointer variant of Scheme [11], a dialect of LISP. To support this interpreter the chip implements an automatic storage allocation system for heap-allocated data and an interrupt facility for user interrupt routines implemented in Scheme. All compound data in the system are built from CAR and CDR pointer list nodes. Each pointer is 32 bits, comprising a 24-bit data field, a 7-bit type field and a 1-bit field used by the storage allocator. The type identifies the object referred to by the data field. This data field is either a literal, or it points to another list node.

The Scheme-79 chip, whose machine organisation is shown in Figure 6 implements a standard von Neumann architecture in which a processor is attached to a memory system. The processor is divided into two parts: the data path and the controller. The data path consists of a set of special-purpose registers, with built-in operators, interconnected by a single 32-bit bus. The controller is a finite-state machine that sequences through the microcode, implementing both the interpreter and garbage collector. At each step it performs an operation on some of the registers (for example, transferring the address in NEWCELL into the STACK register) and selects a next state based on its current state and the conditions developed within the data path.

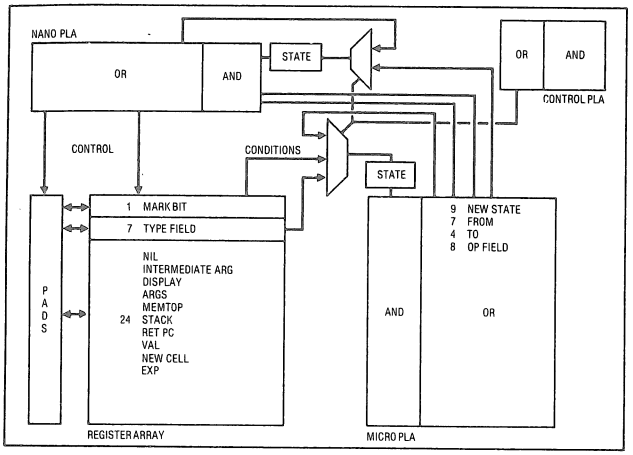


Figure 6: MIT Scheme-79 Machine Organisation

Ten registers in the chip have specialised

characteristics. On each cycle, these registers can be controlled so that one of them is gated onto the bus and selected fields of the bus are gated to another register. For example, VAL holds the value of the last expression, EXP is the register for the current expression, and if this current expression is a procedure call, the arguments are evaluated and built into a list kept in the ARGS register.

The finite-state controller for the Scheme-79 chip is a synchronous system composed of a state register and a large piece of combinational logic - the control map. From the current state stored in the state register, the control map develops control signals for the register array and pads, the new state, and controls for selecting the sources for the next sequential state. The Scheme-79 chip interfaces to the external world through a 32-bit bidirectional data bus that specifies addresses, reads and writes (heap) memory, references I/O devices, reads interrupt vectors, and accesses the internal microcode state during debugging.

The Scheme-79 chip, using a process with a minimum line width of five microns, was 5926 microns wide and 7548 microns long, with a total area of 44.74mm^2. The entire project [11] including prototype tool building and chip synthesis was completed in five weeks.

Berkeley Reduced Instruction Set Computer

The reduced instruction set computer (RISC) [20, 21] design philosophy has been pioneered at the University of California, Berkeley. RISC machines combine a small set of often-used instructions with an architecture which is tailored to their efficient execution. In addition, a single-chip implementation of a simpler machine makes more effective use of the limited resources of present-day VLSI chips - such as the number of transistors, area, and power consumption.

The RISC I [21] and II [20] microprocessors are 32-bit, register-oriented machines designed in nMOS. (RISC I has 138 general-purpose registers, whereas RISC II has 198.) RISC I has 31 operation codes and RISC II has 39, they are mostly simple ALU and shift operations on registers. Both machines use 32-bit addresses and support 8-, 16-, and 32-bit data. They implement two simple addressing modes; indexed and PC-relative; synthesized from these are more complicated addressing modes, most instructions are executed in a simple cycle. The LOAD and STORE instruction - the only operations that access memory - violate this single cycle constraint; they add an index register and the immediate offset during the first cycle, performing the memory access during the next cycle to allow enough time for main-memory access.

Figure 7 shows the machine organization of RISC I. The machine naturally subdivides into the following function blocks: the register-file, the ALU, the shifter, a set of program counter (PC) registers, the data I/O latches, the program status word (PSW) register, and the control section (which contains the instruction register, instruction decoder and internal clock circuits). In addition, the register file needs at least two independent buses because two operands are required simultaneously. In Figure 7, buses A and B are read-only and bus C is write-only.

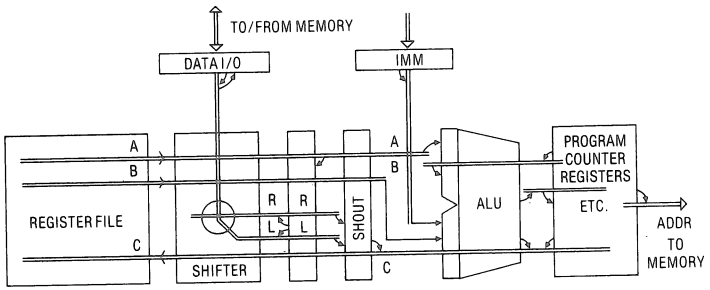


Figure 7: Berkeley RISC Microprocessors

An unusual feature of the two RISC machines is the so-called over lapped window registers, a fast and simple procedure calling mechanism using the general-purpose registers. A procedure has access to 32 register: global (registers 0-9 are shared by every procedure); low (registers 10-15 contain result parameters); local (registers 16-25 are used for local working); and high (registers 26-31 are input parameters). Each time a procedure is called, new registers are allocated in which the low registers of the calling procedure overlap the high registers of the called procedure.

Two different 4-micron nMOS versions of RISC I have been designed; the "gold" version, whose data path is shown in Figure 7, and a "blue" version. Both versions are similar in organization, however a more sophisticated timing scheme in the blue version shortens the machine cycle and reduces chip area. Details of the gold design include a chip size of 406 x 350mm, 44K devices, a design time of 15 man-months, and a layout time of 12 man-months. Power consumption for the chip is estimated at between 1.2 and 1.9 watts. RISC II is 25 percent smaller than RISC I with 41K devices.

4.2. Multi-Processors

General-purpose multi-processor VLSI architectures are constructed from identical simple microcomputers, usually

based on the reduced instruction set philosophy. A microcomputer, containing a primitive processor and a small amount of local on-chip RAM, is able to operate alone as a sequential computer or, with others, as a component of a parallel system.

Two examples are examined below the RIMMS [6] and the innovative INMOS TRANSPUTER [1].

Reduced Instruction Set Multi-Microcomputer System

In the reduced instruction set multi-microcomputer system (RIMMS) [6], illustrated by Figure 8, an attempt was made to keep the design of the component microcomputer as conventional as possible.

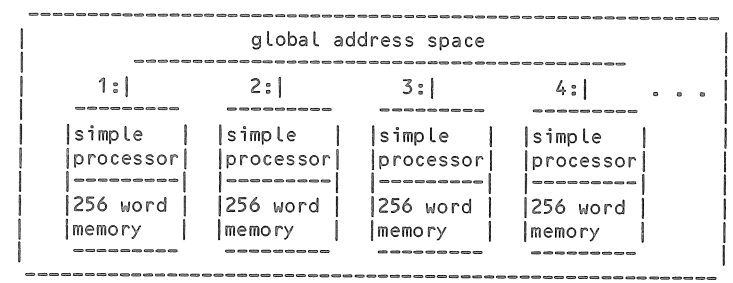


Figure 8: Reduced Instruction Set Multi-Microcomputer System

The central idea in RIMMS is that each microcomputer's local memory behaves as if it is a "bank" of a larger memory. To achieve this, each microcomputer forms part of a global (two-level) address space. An address (16-bits) consists of two parts: the high 8 bits define a specific microcomputer/process, while the low 8 bits define a subsidiary location. Each local memory provides storage for a program fragment (i.e. process) and a process can access any location in the global address space. In addition, a location may be used for "shared memory" or "message passing" communication of data. This is achieved by four memory operations: LOAD, STORE, PUT and TAKE. STORE and LOAD have traditional (shared memory) semantics, while PUT and TAKE support message passing. PUT may only overwrite an "empty" location and TAKE may only remove a non-empty value. Both operations "poll" a location until it is in the correct state.

Although any location in the global address space may be accessed, a process is always executed by its local processor and is executed (atomically) to completion. In

addition, a process can activate another process, using a FORK instruction, as long as its processor is idle. (This atomic execution removes many of the synchronisation problems typically found in control flow multi-processors.) Finally, the processor implementation has a simple, 16-bit von Neumann data path, with only two visible registers (C - program counter, D - base register) holding the state of a process.

The RIMMS chip [6] is a simple, conventional 2-bus data path consisting of a register file, shifter, ALU, MAR/MDR registers, and a control PLA to implement the instruction set. The register file contains 7x16-bit registers and the ALU has two 16-bit input and two output registers. The CPU part of the chip is estimated to be approximately 8mmx8mm in an nMOS process with Lambda = 3 microns.

TRANSPUTER

INMOS' TRANSPUTERS [1] comprise a family of 16- and 32-bit microcomputers, capable of operating alone as a 10-MIPS (million instructions per second) processor or as a component of a parallel network of TRANSPUTERS.

Each microcomputer, as shown below, consists of four main parts: a reduced instruction set processor, 4K bytes of static RAM, a 32-bit multiplexed memory interface, and four INMOS standard serial links providing concurrent message passing to other TRANSPUTERS. The processor has built in support for multi-processing and parallelism. The execution state of each process is defined by six registers. These registers are a three-register evaluation stack, together with an instruction pointer, a workspace pointer, and an operand register. Instructions are eight bits, comprising a 4-bit function code and a 4-bit data value. Operands longer than four bits are built up four bits at a time in the operand register. Basic arithmetic instructions execute in 50 nsecs and a process switch takes only 600 nsecs.

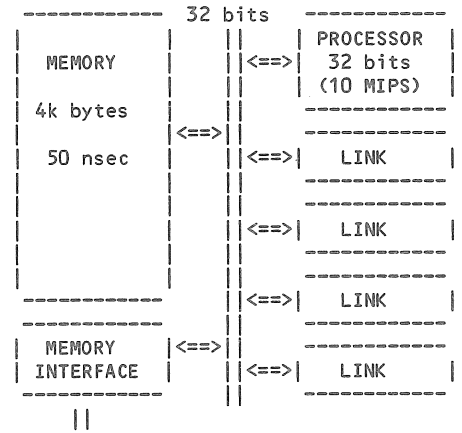


Figure 9: INMOS TRANSPUTER

Communication between TRANSPUTERS is handled by the links. Each link implements two channels, an output and an input, over which messages are transmitted as a series of bytes.

Finally, parallel programming in TRANSPUTERS and its OCCAM programming language [18] is based on communicating processes and message passing using explicitly defined channels. A network of TRANSPUTERS corresponds directly to a network of processes, with each TRANSPUTER supporting one or more processes in a timeshared fashion.

5. FUTURE DIRECTIONS

VLSI microprocessors containing over 100,000 devices are starting to become commonplace [26]. The advantages of placing a sophisticated computer on a chip are well understood to include improvements in its component cost, performance and reliability, together with its power and size. In addition, as the level of integration increases we can expand the chip boundary to encompass cache and main storage in order to improve the performance of the processor. This is advantageous given the relatively slower improvement in the bandwidth available at the pins compared with that in the processor.

A limitation with this "ever larger single processor" approach is that the communications required to execute each instruction must ordinarily traverse all parts of the processor. In making the microprocessor more powerful, additional logic and storage is added, but this increases the

communication costs by increasing the separation of the components. It is clear that such microprocessor designs will reach a point of diminishing cost/performance returns.

In the future improving performance with an ensemble of concurrently operating small processors becomes more attractive than using a large single processor. Thus the INMOS TRANSPUTER and the CMU Programmable Systolic Chip seem to be showing the future direction of VLSI architectures.

REFERENCES

[1] Barron, I., et al, "Transputer does 5 or more MIPS even when not used in parallel," Electronics, vol. 56, no. 23, November 1983, pp. 109-115.

[2] Batali, J., et al, "The SCHEME-81 architecture - system and chip," Proc. Conf. Advanced Research in VLSI , Paul Penfield (ed.), MIT Press, January 1982, pp. 69-77.

[3] Clark, J.H., "A VLSI Geometry Processor for Graphics," COMPUTER, vol. 13, no. 7, July 1980, pp. 59-68.

[4] Conway, L., et al, "MPC79: A Large-Scale Demonstration of a New Way to Create Systems in Silicon," Lambda, 2nd Qtr. 1980, pp. 10-19.

[5] Fisher, A.L., et al, "Architecture of the PSC: A Programmable Systolic Chip," Proc. Tenth Int. Symp. on Computer Architecture, June 1983, pp. 48-53.

[6] Foti, L., et al, "Reduced Instruction Set Multi-Microcomputer System (RIMMS)," Proc. National Computer Conf., July 1984.

[7] Gray, J.P. (ed.), Proc. VLSI 81, Academic Press, New York 1981.

[8] Gupta, A., and Toong, H.D., "An Architecture Comparison of 32-bit Microprocessors," IEEE MICRO, vol. 3, no. 1, February 1983, pp. 9-22.

[9] Guterl, F., "Microprocessors Chip Architecture: a revolution brewing," IEEE Spectrum, July 1983, vol. 20. no. 7, pp. 30-37.

[10] Hennessy, J.L., "VLSI Processor Architecture," IEEE Trans. on Computers, December 1984, vol. C-33, no. 12, pp. 1221-1246.

[11] Holloway, J., et al, "SCHEME-79 - LISP on Chip," COMPUTER, vol. 14, no. 7, July 1981, pp. 10-21.

[12] Jansen, W.D., and Fairbairn, D.G., "The Silicon Foundry: Concepts and Reality," Lambda, 1st Qtr. 1981, pp. 16-26.

[13] Kung, H.T., "Why Systolic Arrays," COMPUTER, vol. 15, no. 1, January 1982, pp. 37-46.

[14] Kung, R.F., et al (ed.), Proc. CMU Conf. VLSI Systems and Computations, Springer-Verlag, New York, October 1981.

[15] Lepselter, M.P., "High Speed Silicon Integrated Circuits Using X-ray Lithography," Conf. Proc. on Advanced Research in VLSI, MIT, January 25-27, 1982.

[16] Lyon, R.F., "A Bit-Serial VLSI Architecture Methodology for Signal Processin," Proc. VLSI 81, August 1981, pp. 131-140.

[17] Lyon, R.F., "The Optical Mouse, and an Architecture Methodology for Smart Digital Sensors," Technical Report VLSI-81-1, Xerox Corporation Palo Alto Research Center, August 1981.

[18] May, D., "OCCAM," ACM SIGPLAN Notices, vol. 18, no. 4, April 1983, pp. 69-79.

[19] Mead, C.A., and Conway, L., "Introduction to VLSI Systems," Addison-Wesley, Reading, Mass., 1980.

[20] Patterson, D.A., "Reduced Instruction Set Computers," Comm. ACM, vol. 28, no. 1, January 1985, pp. 8-21.

[21] Patterson, D., and Sequin, C., "A VLSI RISC," COMPUTER, vol. 15, no. 9, September 1982, pp. 8-21.

[22] Penfield, P. (ed.), 1982 Conf. on Advanced Research in VLSI, MIT January 1982.

[23] Rivest, R.L., "A Description of a Single-Chip Implementation of RSA Cipher," Lambda, 4th Qtr. 1980, pp. 14-18.

[24] Seitz, C., "Ensemble Architectures for VLSI - A Survey and Taxonomy," Proc. 1982 Conf. on Advanced Research in VLSI, P. Penfield (ed.), MIT, January 1982, pp. 33-45.

[25] Seitz, C.L., "Concurrent VLSI Architectures," IEEE Trans. on Computers, December 1984, vol. C-33, no. 12, pp. 1247-1265.

[26] Treleaven, P.C., "VLSI Processor Architectures," COMPUTER, vol. 15, no. 6, June 1982, pp. 33-45.

V CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO
XI CONFERÊNCIA LATINO-AMERICANA DE INFORMÁTICA

COMISSÃO ORGANIZADORA

PRESIDENTE DO CONGRESSO: Claudio Zamitti Mammana

COMISSÃO PROMOTORA:

Prof. Francisco Luis dos Santos Ferraz - **Reitor da UFRGS**
Prof. Claudio Zamitti Mammana - **Presidente da SBC**
Prof. Jorge Vidart - **Presidente do CLEI**

COORDENAÇÃO GERAL:

Manoel Agamemnon Lopes - **SBC**
Aldo Migliario Osório - **CLEI**

COMISSÃO EXECUTIVA:

Coordenador:

Simão Sirineo Toscani	UFRGS
Carlos Ernesto Rech	UFRGS
Cláudia Sabani	UFRGS
Clésio Saraiva dos Santos	UFRGS
Helena Vaythier de Souza	UFRGS
Luiz Julião Braga Filho	UFViçosa
Paulo R.F. Mosca	UFRGS
Philippe O.A. Navaux	UFRGS
Ricardo A.L. Reis	UFRGS
Roberto M.J. Macedo	UFRGS
Sergio Felipe Zirbes	UFRGS
Therezinha da Costa Chaves	UFRJ

SECRETARIA EXECUTIVA:

Lourdes Tassinari	UFRGS

COMISSÃO DO SEMISH:

Coordenação Geral:

Ricardo Augusto da Luz Reis	UFRGS

Comissão Nacional:

Coordenador:

Ricardo Augusto da Luz Reis	UFRGS
Antonio Carlos R. Costa	UFRGS
Artur João Catto	UFSCar
Leo Pini Magalhães	UNICAMP
Merval Jurema Filho	UFPe
Michael Stanton	PUC/RJ
Newton Faller	UFRJ
Roberto da Silva Bigonha	UFMG
Siang Wun Song	USP

Comissão Latino-Americana
Coordenador
Roberto M.J. Macedo
Universidade Federal do Rio Grande do SUL - **Brasil**

Hector Rodrigues Estay
Universidade Católica de Valparaiso - **Chile**

Juan M. Pereira
Universidad Simón Bolivar - **Venezuela**

Leonardo Elizalde
Pontifícia Univ. Católica de Equador - **Equador**

Leopoldo Carranza
Sadio - **Argentina**

Oscar Herreros
Universidad Católica de Assunción - **Paraguai**

Simon M. Tenzer
Universidad de la República - **Uruguai**

Tulio Galvez
Associación Peruana de Computación e Informática - **Peru**

Victor Toro
Universidad de los Andes - **Colombia**

Grigot Ponce de Leon
Universidade Privada de Santa Cruz - **Bolivia**

COMISSÃO DO SBCCI

Coordenador:

Ricardo Augusto da Luz Reis	CPGCC/UFRGS
Altamiro Amadeu Suzim	CPGCC/UFRGS
Ana Maria Kuniyoshi	IM/CTI
Antonio Carlos Barbosa	ITAUCOMP
Carlos Ignácio Mammana	IM/CTI
Eber Assis Schmitt	NCE/UFRJ
Jean Albert Bodinaud	IME/USP
Mario Vaz da Silva	DEE/UFRJ
V ictor Blatt	CPqD/TELEBRÁS

COMISSÃO DO SECOMU

Coordenador

Henriqu e Paca Loureiro Luna	UFMG

COMISSÃO DO JAI

Coordenadora:

Lidia Michaela Segre	COPPE/UFRJ

COMISSÃO DO CTIC

Coordenadora:

Alaide Mammana	UNICAMP
Paulo Alberto de Azeredo	UFRGS
Lilian Markenson	COPPE/UFRJ
Mario Ferraretto	CTI
Newton Faller	UFRJ

COMISSÃO DA EXTEC

Coordenador:

Tiaraju Wagner	UFRGS
Carlos Ernesto Rech	UFRGS
Rejane Telichevesky	UFRGS

COMISSÃO CULTURAL

Coordenadora:

Cláudia Sabani	UFRGS
Helena Vauthier de Souza	UFRGS
Cleto José Beuren	UFRGS

COMISSÃO DE EDIÇÃO

Coordenador:

Carlos Ernesto Rech	UFRGS
Nicolau Meisel	PUC/RJ

CENTRO LATINO-AMERICANO DE ESTUDOS EM INFORMÁTICA